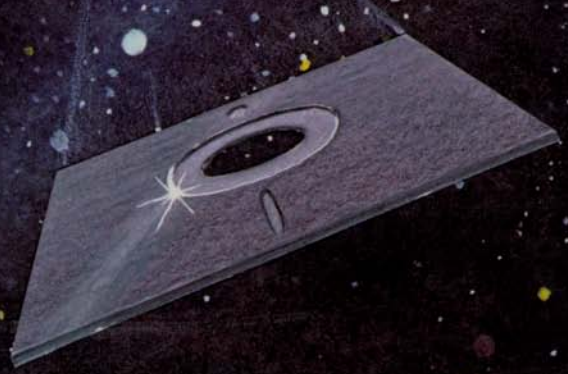
Best of INTERFACE AGE

VOLUME I SOFTWARE IN BASIC

Edited by carl d. warren



TINO ORTIZ

Best of Interface Age

Volume 1: Software in BASIC

Best of Interface Age

Volume 1: Software in BASIC

Carl D. Warren
Editor-in-Chief

dilithium Press
Beaverton, Oregon

©Copyright, Interface Age, 1979

10 9 8 7 6 5 4 3

All rights reserved. No part of this book may be reproduced in any form or by any means without permission in writing from the publishers, with the following two exceptions: any material may be copied or transcribed for the non-profit use of the purchaser; and material (not to exceed 300 words and one figure) may be quoted in published reviews of this book.

ISBN: 0-918398-36-3
Library of Congress catalog card number: 79-67462

Printed in the United States of America.

dilithium Press
11000 SW 11th St.
Beaverton, Oregon 97005

Preface

Several months ago, it became apparent that we should give serious thought to creating a "best of" series from the pages of *Interface Age*. The idea seemed like it would not present a major problem at first, until we looked at the mountain of material we had available. The quantity wasn't the only problem. The decision had to be made on what was best, so it could be included in the reprint series.

After careful consideration, with close attention being paid to readers' letters, the decision was made to create a "best of" series of *Interface Age* Classics. What this meant was that rather than to create a huge, difficult to use book of just about everything, we are printing the classic software pieces and the ones that many readers said they would like to see.

The four reprinted articles that you find in this volume are those classics. Furthermore, they represent several different programming techniques. At the same time, they provide the reader with some of the most useful software ever created.

Chapter four, "The Great Experiment," offers the one piece of software that has been requested more than any other: the complete source listing of Uiterwyk's 4K BASIC. It is an absolute classic. The source code was released into the public domain by Mr. Uiterwyk for this software volume, so students of software could achieve maximum benefit from its use. For this, we at *Interface Age* are extremely grateful.

My staff and I are quite convinced that everyone who makes use of this first volume will find it invaluable in making better use of their systems.

Volume 2 of this series is made up of additional general purpose software from some of the *Interface Age* authors you have enjoyed over the past few years. Volumes 3 and 4 are dedicated specifically toward the small businessman. These two volumes contain more business software than any book currently available today.

To round out this current "best of" series, Volume 5, *Best of Interface Age—Things to Think About*, contains those articles for the futuristic thinker and gadgeteer. This volume contains reprints from Roger Garret's famous "Inventor's Sketchpad." Also included are short and useful software tips, along with some handy hardware articles.

Interface Age has always been known as the leader in the publication of important and useful software and ideas. We feel that the publication of this "best of" series will further enhance our leadership and provide you with hundreds of hours of enjoyment.

carl warren
editor-in-chief

Acknowledgment

When a reprint series is created, it is of utmost importance to thank the authors whose works are being reprinted. We gladly extend our appreciation to them. We also thank all those authors that contributed articles over the past forty-plus issues, and the readers that have made us such a success.

Robert S. Jones
Publisher

Table of Contents

Chapter 1 Lawrence Livermore 8080 BASIC 1
by Jerry Barber, Royce Eckard, John Dickenson, David Mead, and E. R. Fisher
Original publication date: December 1976, January, February, and March 1977

Chapter 2 Dr. Wang's Palo Alto TINY BASIC 139
by Roger Rauskolb
Original publication date: December 1976

Chapter 3 National's TINY BASIC—NIBL 171
by Phil Roybal and Mark Alexander
Original publication date: December 1976, January 1977

Chapter 4 The Great Experiment—Floppy ROM™ # 1 Robert Uiterwyk's 6800 4K BASIC 229
by Bill Turner and William Blomgren
Original publication date: May, July 1977

Appendix A General Software Index 301
A comprehensive index of general purpose software printed in *Interface Age* since January, 1977

Appendix B Available Back Issues 311
A list of all the back issues that are still available, and how to obtain them.

Index 313

Chapter 1

Lawrence Livermore Laboratories 8080 Basic Interpreter

by Jerry Barber, Royce Eckard John Dickenson, David Mead and E. R. Fisher

The year 1976 saw a number of accomplishments to the microcomputer world, which was still very much in its infancy. The most important developments were with BASIC interpreters being made available to the micro user.

One such BASIC, was the 8080 version developed at the University of Idaho by John Dickenson, Jerry Barber, and John Teeter; under a contract with the Lawrence Livermore Laboratory. The floating point package was developed by David Mead, modified by Hal Brand and Frank Olken.

The entire project, including the course listing of the intérpreter was documented in the December 76, January 77, February 77, and March 77 issues of Interface Age magazine.

Because BASIC is still considered the most important language in the microcomputer industry, and also due to the continuing interest in the LLL version, we have re-published it here in its entirety.

STORAGE REQUIREMENTS

The BASIC interpreter consists of a 5K-byte-PROM resident interpreter used for program generation and debug was configured to operate with the MCS-8080 microprocessor.

The goal in developing the 8080 BASIC was to provide a high-level, easy-to-use conversational language for performing both control and computation functions in the MCS-8080 microprocessor. To minimize system memory size and cost, the interpreter was constrained to fit

into 5K bytes. It was necessary, therefore, to limit the commands to those considered the most useful in microprocessor applications.

MATH OPERATOR EXECUTION TIMES

Average execution times of the four basic math operators are as follows:

Operation	Execution time on 8080 (m sec)
ADD	2.4 m sec
SUBTRACT	2.4 m sec
MULTIPLY	5.4 m sec
DIVIDE	7.0 m sec

BASIC INTERPRETER LANGUAGE GRAMMAR

COMMANDS—Six BASIC interpreter commands are provided. These commands are:

RUN	Begins program execution
SCR	Clears program from memory
LIST	Lists ASCII program in memory
PLST	Punches paper-tape copy of program
PTAPE	Reads paper-tape copy of program using high-speed reader
CNTRL S	Interrupts program during execution

The LIST and PLST commands can be followed by one or two line numbers to indicate that only a part of the program is to be listed. If one line number follows the command, the program is listed from that line number to the end of the program. If two line numbers (separated by a comma) follow the command, the listing begins at the first line number and ends at the second.

When a command is completed, READY will be typed on the teletype. Once initialized by a command, a process will normally go to completion. However, if you wish to interrupt an executing program or a listing, simply strike CNTRL S and the process will terminate and a READY message will be typed.

STATEMENTS—Each statement line begins with a line number, which must be an integer between 0 and 32767. Statements can be entered in any order, but they will be executed in numerical order. All blanks are ignored. The following types of statements are allowed:

REM—Indicates a remark (comment). The system deletes blanks from all character strings that are not enclosed in quotes ("). Therefore, it is suggested that characters following the REM key word be enclosed in quotes.

END—Indicates the end of a program. The program stops when it gets to the END statement. All programs must end with END.

STOP—Stops the program. This statement is used when the program needs to be stopped other than at the end of the program text.

GOTO—Transfers program control to specified statement line number. This statement is used to loop or jump unconditionally within a program. Program execution continues from new statement.

DIM—Declares an array. Only one-dimensional arrays with an integer constant number of elements are allowed. An array with N elements uses indexes 0 and N-1. All array locations are set to zero. No check is made on subscripts to ensure that they are within the declared array. An array variable must be a single letter.

LET—Indicates an assignment statement (Addition, subtraction, multiplication, division, or special function may be used). The LET statement is used to assign a value to a variable. Non-array variables can be either a single letter or a letter followed by a digit. It is possible to have an array and a non-array variable with the same name. The general form of the LET statement is:

line number LET identifier = expression,

where *identifier* is either a subscripted array element or a non-array variable or function and *expression* is a unary or binary expression. The expression will be one of the following ten types:

variable
– variable
variable + variable
variable – variable
– variable + variable
– variable – variable
variable * variable
– variable * variable
variable / variable
– variable / variable,

where *variable* is an identifier, function, or number. The subscript of an array can also be an expression.

IF—Condition statement which transfers to specified line number statement if the condition of the expression is met. It has the form: line number IF expression relation expression THEN transfer line number. The possible relations are:

Equal	=
Greater than	>
Less than	<
Greater than or equal	>= =<
Less than or equal	<= =<
Not equal	<> ><

If the relation between the two expressions is true then the program transfers to the line number, otherwise it continues sequentially.

INPUT—This command allows numerical data to be input via the teletype. The general form is:

Line number INPUT identifier list,

where an *identifier list* is a sequence of identifiers separated by commas. There is no comma after the last identifier so, if only one identifier is present, no comma is needed. When an INPUT statement is executed, a colon (:) is output to the teletype to indicate that data are expected. The data are entered as numbers separated by commas. If fewer data are entered than expected, another colon is output to the teletype, indicating again that data are expected. For example, where

50 INPUT I,J,K,P

is executed, a colon is output to the teletype. Then, if only 3 numerical values are entered, another colon will be output to indicate that more data are expected; e.g.,

:4,4,6.2 C/R
:10.3 C/R

where C/R is the carriage-return key. If an error is made in the input-data line, an error message is issued and the entire line of data must be reentered. If, for the above example,

:4,4,6M2,10.3 C/R

is entered, the system will respond:

INPUT ERROR, TRY AGAIN
:

At this time, the proper response would be

4,4,6.2,10.3 C/R.

PRINT—This command allows numerical data and character strings to be printed on the teletype. Two types of print items are legal in the print statement: character strings enclosed in quotes (") and expressions. These items are separated by either a comma or a semicolon. If print items are separated by a comma, a skip occurs to the next pre-formatted field before printing of the item following the comma begins. The pre-formatted fields begin at columns 1, 14, 27, 40, and 52. If print items are separated by a semicolon, no skip occurs. If a semicolon or comma is the last character on a print statement line, the appropriate formatting occurs and the carriage-return-line feed is suppressed. A print statement of the form

50 PRINT

will generate a carriage-return-line feed. Thus the two lines below

50 PRINT "INPUT A NUMBER";
60 INPUT A

will result in the following output:

INPUT A NUMBER:

FOR—Causes program to iterate through a loop a designated number of times.

NEXT—Signals end of loop at which point the computer adds the step value to the variable and checks to see if the variable is still less than the terminal value.

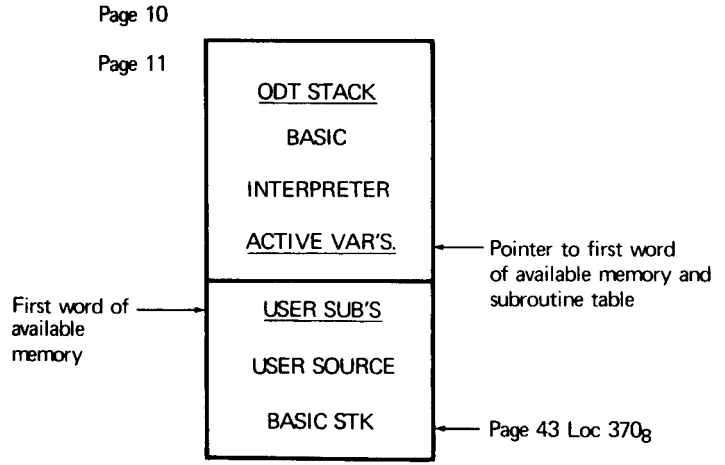
GOSUB—Transfer control to a subroutine that begins at specified line number.

RETURN—Returns control to the next sequential line after the last GOSUB statement executed. A return statement executed before GOSUB is equivalent to a STOP statement.

CALL—Calls user-written assembly-language routines of the form

CALL (N,A,B,. . .),

where N is a subroutine number from 0-254 and A, B,. . . are parameters. The parameters can be constants, variables, or expressions. However, if variables and constants or expressions are inter-mixed, all variables should have been referenced before the CALL statement. Otherwise, the space reserved for newly referenced variables may overwrite the results of constants and expressions. A memory map of one configuration of the system is shown below.



The subroutine table contains 3-byte entries for each subroutine. The table directly follows the pointer to the first word of available memory (FWAM) and must end with a octal 377. A sample table and its subroutines is shown below:

```
        ORG 16612Q
DWSUBEND    ;Define FWAM
        DB 1        ;Subroutine #1
        DW SUB1     ;Starting add of subroutine
                    #1
        DB 4        ;Subroutine #4
```

```
DW SUB4          ;Starting add of subroutine
                 #4
DB 5             ;Subroutine #5
DW SUB5          ;Starting add of subroutine
                 #5
DB 2             ;Subroutine #2
DB SUB2          ;etc.
DB 377Q          ;end of subroutine table
SUB1:↓           ;Subroutine #1
     RET
SUB5:↓           ;Subroutine #5
     RET
     •
     •
     •
     RET         ;Retain last subroutine
SUBEND EQU$      ;FWAM
```

Addresses to passed parameters are stored on the stack. The user must know how many parameters were passed to the subroutine. These must be taken off the stack before RET is executed. Addresses are stored last parameter first on the stack. Thus, on entry to a subroutine, the first POP instruction will recover the address to the last parameter in the call list. The next will recover the next to last, etc.

Each scalar variable passed results in the address to the first byte of a four-byte block of memory. Each array element passes the address to the first byte of a (N-M) × four-byte memory block, where N is the number of elements given the array in the DIM STMT and M is the array subscript in the CALL STMT.

For passed parameters to be handled in expressions within BASIC, they must be in the proper floating-point format.

FUNCTIONS—Two special functions not found in most BASIC codes are available to input or output data through Intel 8080 port numbers. These functions are;

GET (X) = READ 8080 INPUT PORT X.
PUT (Y) = OUTPUT A BYTE OF DATA TO OUTPUT PORT Y.

The function GET allows input from a port and the function PUT allows output to a port. Their general forms are:

GET (expression).
PUT (expression).

The function GET may appear in statements in a position that implies that a numerical value is used. The function PUT may appear in

statements in a position that implies that a numerical value will be stored or saved. This is because GET inputs a number and PUT outputs a number. For example, while

LET PUT(I) = GET(J) is valid

LET GET(I) = PUT(J) is invalid.

These functions send or receive one byte of data, which in BASIC is treated as a number from 0 to 255.

VARIABLES—Single characters A→Z

Single character followed by a signal decimal digit

NUMBERS—Numbers in a program statement or input via the teletype are handled with a floating-point package provided by LLL. Numbers can have any of the following forms:

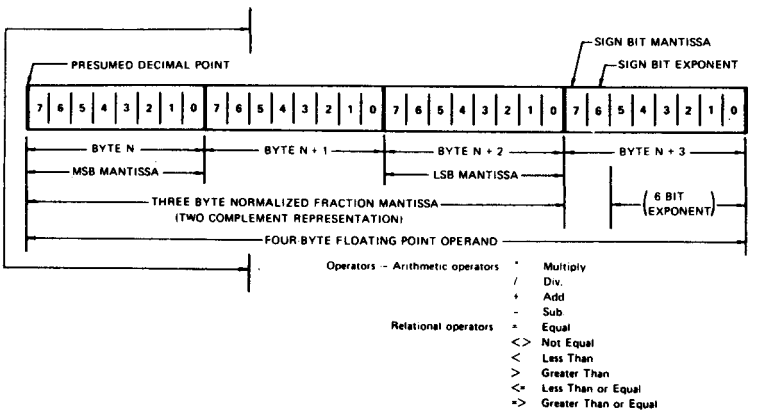
4	±4.	.123
4.	±4.0	±.123
4.0	1.23	0.123
±4	±1.23	±0.123

and the user may add an exponent to any of the above forms using the letter E to indicate powers of 10. The forms of the exponent are:

E±1	E±15
E 1	E 15
E 1	E 15

The numbers are stored with seven-digit accuracy; therefore, seven significant figures can be entered. The smallest and largest numbers are ±2.71051E-20 and ±9.22337E18.

Floating point numbers are expressed as a 32 bit operand consisting of a 24 bit normalized fractional mantissa in standard two's complement representation and a 6 bit exponent also is standard two's complement representation with a range of −64 to +63. The exponent byte also includes the exponent sign bit and mantissa sign bit. The floating point number format is as shown in the following:


PRESUMED DECIMAL POINT
SIGN BIT MANTISSA
SIGN BIT EXPONENT
7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0
BYTE N
BYTE N + 1
BYTE N + 2
BYTE N + 3
MSB MANTISSA
LSB MANTISSA
THREE BYTE NORMALIZED FRACTION MANTISSA
(TWO COMPLEMENT REPRESENTATION)
6 BIT EXPONENT
FOUR BYTE FLOATING POINT OPERAND
Operators – Arithmetic operators
* Multiply
/ Div.
+ Add
- Sub
Relational operators
= Equal
<> Not Equal
< Less Than
> Greater Than
<= Less Than or Equal
=> Greater Than or Equal

INTERPRETER OPERATION

INITIALIZATION—The BASIC interpreter is presently configured so that it is located in memory pages 11_8 to 34_8. The starting address is page 17_8, location 0. This address begins an initialization sequence that allows the user to begin with a clear memory. However, to avoid the initialization sequence, a second starting address—page 17_8 to 34_8—can be used. This starting address is used if the user wishes to retain any program that might exist in memory.

Once started the interpreter responds with READY.

INPUT LINE FORMAT

Each line entered is terminated with the carriage-return key. The line-feed key is ignored. Carriage-return automatically step terminal to next line and waits for next line statement number input. Statements can be entered in any order, but they will be executed in numerical order. All blanks outside of quotation marks are ignored by the interpreter. Up to 72 characters may be entered/line.

INPUT LINE EDITING—A program can be edited by using the line numbers to insert or delete statements. Typing a line number and then typing a carriage return causes the statement at that line number to be deleted. Since the statements can be entered in any order, a statement can be inserted between two existing statements by giving it a line number between the two existing statement line numbers. To replace a statement, the new statement should have the same line number as the old statement.

It is possible to correct errors on a line being entered by either deleting the entire line or by deleting one or more characters on the line. A character is deleted with either the rubout key or the shift/O key. Several characters can be deleted by using the rubout key several times in succession. Character deletion is, in effect, a logical backspace. To delete the line you are currently typing, use the CNTRL/Y key.

BASIC PROGRAM EXECUTION—Entering a RUN command, after a BASIC program has been entered into the microcomputer, will cause the current program to begin execution at the first statement number. RUN always begins at the lowest statement number.

ERROR MESSAGES—If an unrecognizable command is entered, the word WHAT? is printed on the teletype. Simply retype the command. It may also have been caused by a missing line number on a BASIC statement, in which case you should retype the statement with a line number.

During program execution and whenever new lines are added to the program, a test is made to see if there is sufficient memory. If the memory is full, MEMORY FULL is printed on the teletype. At this point, you should enter one of the single digits below to indicate what you wish to do:

Number Entered	Meaning
0	(RUN) runs
0	(RUN) runs the program in memory
1	(PLST) outputs program in memory to paper tape punch
2	(LIST) lists program in memory
3	(SCR) erases program in memory
4	none of the above (will cause WHAT? to be printed out on the teletype).

To help you select the best alternative, a brief description of how the statements are manipulated in memory will be helpful. All lines entered as program are stored in memory. If lines are deleted or replaced, the originals still remain in memory. Thus, it is possible, if a great deal of line editing has been done, to have a significant portion of memory taken up with unused statements. If a MEMORY FULL message is obtained in these circumstances, then the best thing to do is punch a tape of the program (entering number 1), then erase the program memory with a SCR command (or a number 3, if memory is too full to accept commands), and then re-enter your program using the high-speed paper-tape reader with the PTAPE command.

If an error is encountered while executing a program, an error message is typed out that indicates an error number and the line number in which the error occurred. These numbered error messages are as follows:

Error Number	Error Message
1	Program has no END statement
2	Unrecognizable keyword at beginning of statement
3	Source statements exist after END statement
4	Designation line number is improperly formed in a GOTO, GOSUB, or IF statement
5	Designation line number in a GOTO, GOSUB, or IF statement does not exist
6	Unexpected character
7	Unfinished statement
8	Illegally formed expression
9	Error in floating-point conversion
10	Illegal use of a function
11	Duplicate array definition
12	An array is referenced before it is defined
13	Error in the floating-point-to-integer routine, Number is too big
14	Invalid relation in an IF statement

LLL 8080 BASIC INTERPRETER PROGRAMS EXAMPLES

PRINT STATEMENT PROGRAM EXAMPLE—The program below gives a few examples of the use of the print statement.

```
1 PRINT"THE PRE-FORMATTED COLUMNS ARE SHOWN BELOW"
2 PRINT 1,2,3,4,5
4 PRINT
10 PRINT "INPUT 1ST NUMBER"
20 INPUT A
30 PRINT "INPUT 2ND NUMBER"
40 INPUT B
50 PRINT
60 PRINT"A IS";A
70 PRINT"B IS";B
80 PRINT"A IS";A;"B IS";B;"A+B IS";A+B
100 END
```

```
THE PRE-FORMATTED COLUMNS ARE SHOWN BELOW
 1      2      3      4      5

INPUT 1ST NUMBER
 2
INPUT 2ND NUMBER
 3

A IS 2
B IS 3
A IS 2 B IS 3 A+B IS 5
```

PLOT FUNCTION PROGRAM—The following program plots a function on a display. It uses four user-written assembly-language subroutines. The display works as follows: The contents of memory locations on pages 274_8 to 277_8 are displayed as 16 rows of 64 characters each. Thus, if location 201_8 on page 274 contains 301_8 (ASCII A), an A appears in column 2 of Row 3. An example of this program's execution is shown below:

```
RUN
WHAT SHOULD PLOT BE LABELED? MCS80—BASIC INTERPRETER
READY
```

The BASIC and assembly-language programs and the display output are shown below.

BASIC PROGRAM

Display output for Plot Function program.

```
1 REM" THIS ROUTINE WILL PLOT A SET OF AXIS AND QUADRATIC FUNCTION
2 REM" ON A DISPLAY AND THEN LABEL IT.  IT USES A 4 USER WRITTEN
3 REM" SUBROUTINES:
4 REM
5 REM" CALL(1,X,Y,C) - PLACES C IN COLUMN X, ROW Y OF THE DISPLAY
6 REM" WHERE C IS AN ASCII CODED CHARACTER
7 REM
8 REM" CALL(2,A(0)) - READS A CHARACTER STRING FROM THE TTY AND STORES
```

```
9 REM" IT IN ARRAY A
10 REM
11 REM" CALL(3,A(0)) - WRITES THE CHARACTER STRING STORED IN ARRAY A
12 REM" TO THE DISPLAY
13 REM
14 REM" CALL(4) - CLEARS THE DISPLAY
15 REM
16 REM" START OF PROGRAM
17 REM
18 REM" RESERVE STORAGE AREA FOR TITLE
20 DIMA(10)
30 REM" CLEAR SCREEN
40 CALL(4)
50 REM" ASK FOR AND INPUT TITLE
55 PRINT"WHAT SHOULD PLOT BE LABELED ?";
60 CALL (2,A(0))
70 REM" DRAW AXIS
80 GOSUB 500
90 REM" PLOT FUNCTION
100 LET X= -29
110 GOSUB 1000
120 CALL (1,31+X,8-Y,248)
130 LEXTX = X+1
140 IF X><31 THEN 110
150 REM" OUTPUT TITLE
160 CALL (3,A(0))
165 REM" WE'RE DONE
170 STOP
500 REM" THIS SUB. WILL DRAW A SET OF AXIS
505 LET X=1
510 LET Y=7
520 LETC=173
530 CALL (1,X,Y,C)
540 LETX=X+1
550 IF X><65 THEN 530
560 LET X=31
570 LET Y=1
575 LECTC=252
580 CALL (1,X,Y,C)
590 LETY=Y+1
600 IF Y>< 17 THEN 580
610 RETURN
1000 REM" GIVEN X THIS SUB. CALCULATES (17/900)*X**2-8
1005 REM" FIRST CHECK IF X=0 AS IT WILL UPSET FLOATING POINTPACKAGE
1010 IF X=9 THEN 1045
1015 REM" WE'RE OK - CALCULATE FUNCTION
1020 LET Y=X*X
1025 LET K=17/900
1030 LET Y=Y*K
1035 LET Y=Y-8
1040 RETURN
1045 LETY=-8
1050 RETURN
2000 END
```

```
          .TITLE   'ASSEMBLY LANGUAGE PROGRAM'
          .LOC     16614Q
          ;        DEFINE EXTERNALS
 FIX      EQU      14012Q   ;        FIX ROUTINE
 COPDH    EQU      13212Q   ;        COPY ROUTINE
 FREG1    EQU      16567Q   ;        FLOATING POINT REGISTER
          DW       SBEND    ;        FWAM
          ;        ENTRIES IN SUB TABLE
          DB       1
          DW       SCOPE

          DB       2
          DW       SUB2
          DB       3
          DW       SUB3
          DB       4
          DW       SUB4
```

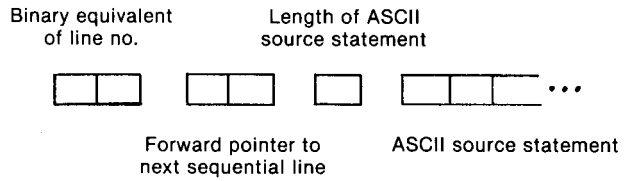
```
        DB      377Q'   ;       NO MORE ENTRYS
        ;       THE CALL TO THIS ROUTINE IS OF THE FORM
        ;               CALL (1 X Y C)
        ;       THE VALUE OF C IS PLACED IN COLUMN X LINE Y
        ;       OF THE DISPLAY

SCOPE:  POP     D       ;       ADDRESS OF CHARACTER
        LXI     H,FREG1 ;       COPY TO FREG1
        CALL    COPDH
        XCHG            ;       ADDRESS TO DE
        CALL    FIX     ;       FIX IT
        INX     D
        INX     D       ;       POINT TO 4TH BYTE
        INX     D
        LDAX    D       ;       GET CHARACTER
        MOV     B,A     ;       SAVE IT IN B
        POP     D       ;       ROW ADD
        LXI     H,FREG1 ;       COPY TO FREG1
        CALL COPDH
        XCHG
        CALL    FIX     ;       FIX IT
        INX     D       ;       GET BYTE 4 TO A
        INX     D
        INX     D
        LDAX    D
        MOV     C,A     ;       SAVE IN C
        POP     D       ;       GET COLUMN ADD
        LXI     H,FREG1 ;       COPY TO FREG1
        CALL    COPDH
        XCHG
        CALL    FIX     ;       FIX IT
        INX     D       ;       POINT TO 4TH BYTE
        INX     D
        INX     D
        LDAX    D       ;       GET IT TO A
        ;       CALCULATION OF ADDRESS
        LXI     H,135777Q

        LXI     D,100Q
LUP:    DCR     C
        JZ      ADINC
        DAD     D
        JMP     LUP
ADINC:  MOV     E,A
        DAD     D       ;       ADD IN COLUMN LOC
        MOV     M,B     ;       STORE CHARACTER
        RET             ;       DONE
        ;       SUB2 READS A TITLE FROM TTY VIA ODT
READ    EQU     333Q    ;       ODT ROUTINE
SUB2:   POP     H       ;       GET STORAGE AREA ADD
        PUSH    H
        MVI     C,0     ;       INIT CNTR

LUP2:   INX     H       ;       BUMP PNTR
        CALL    READ    ;       READ A CHARACTER
        CPI     215Q    ;       CR?
        JZ      DUN2    ;       YES - DONE
        INR     C       ;       INCR CNTR
        MOV     M,A     ;       SAVE CHARACTER
        JMP     LUP2
DUN2:   POP     H
        MOV     M,C     ;       STORE CNT
        MVI     X,212Q  ;       SEND A LF
        RST     6
```

```
        RET             ;       DONE
        ;       SUB3 WRITES TITLE TO DISPLAY
SUB3:   POP     H       ;       GET ADD
        LXI     D,137741Q
        MOV     C,M
        INX     H
LUP3:   MOV     A,M     ;       SEND STRING
        STAX    D
        INX     H
        INX     D
        DCR     C
        JNZ     LUP3
        RET
        ;       SUB4 CLEARS SCREEN
SUB4:   LXI     H,136000Q
        MVI     A,240Q  ;       SPACE
        MVI     D,0     ;       CNTR S
        MVI     C,4
LUP4:   MOV     M,A     ;       CLEAR IT
        INX     H
        DCR     D
        JNZ     LUP4
        DCR     C
        JNZ     LUP4
        RET
SBEND   EQU     $
        .END
```

DESCRIPTION OF THE INTERPRETER

Following is a brief description of the BASIC interpreter. Hopefully, with this description, it will not be a major project to modify the BASIC to satisfy the reader's specific needs.

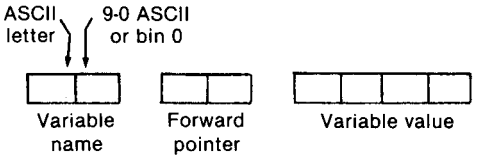
Formats—Source statements are stripped of blanks on input (character strings enclosed in quotes ["] are an exception) and stored as is in memory, using the following format:


Binary equivalent of line no.
Length of ASCII source statement
Forward pointer to next sequential line
ASCII source statement

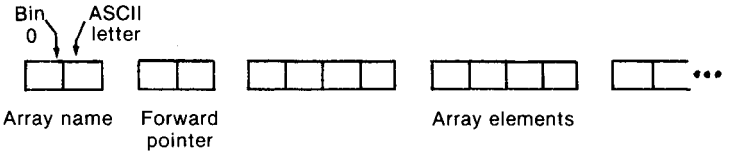
The forward pointer links statements by ascending line numbers. The last line's forward pointer (supposedly an end statement) has value 177777_8 to indicate end of the list.

The symbol table is built up at run time and begins after the most recently entered source statement (the variable STSPAC points to where the symbol table will start). Symbol table entries are shown on the next page.

SCALAR-VARIABLE FORMAT



ARRAY-VARIABLE FORMAT



Subroutines—Following is a list of potentially useful subroutines, with a brief description of each subroutine:

ALPHA — Value pointed to by H and L is tested to see if it is an ASCII Letter.
CY = 1 = >Yes
CY = 0 = >No

NUMB — Same as above but tests for a decimal number (ASCII 0-9).

CHAR2 — Inputs a character from the teletype to a register.

CHK1 — Checks to see if HL are equal to 177777_8 (−1).
CY = 1 = >Yes
CY = 0 = >No

CONV (CVRT) — One of the floating-point routines. Converts floating-point number to a character string. Output is padded to the output buffer.

COPDH — Copies floating-pont number pointed to by D, E to location pointed to by H, L; uses copy.

COPY — One of the floating-point routines. Copies floating-point value pointed to by A, L to location pointed to by H, C.

CUB — Converts the integer-character string pointed to by H, L to its binary equivalent. Vale returns in D, E registers.

DCOMP — Double-byte comparison routine. Compares value in CB to the value in ED.
Z = 1 => CB = ED
CY = 1 => CB>ED
CY = 0 => CB≤ED

DFXL — One of the floating-point routines. Used to float an unsigned integer H, L point to first of four bytes; integer is right justified in first three bytes.

EVAL — Evaluates an expression the first element of which is pointed to by H, L and the length of which is in C. Used to evaluate expressions wherever they are legal in BASIC. C usually contains the length of the source statement line containing the expression.

FINPT — One of the floating-point routines. Converts character string to floating-point number. The variable HLINP contains a pointer to the character string, and the variable CREG contains the length of line containing character string. Mode = 0 => data comes from teletype (i.e., only delimiters are g's). Mode = 1 => data comes from source statements.

FIX — Fixes a floating-point number. DE points to number to be fixed. Error code 13 is given if number is too big to fix.

FSYM — Finds symbols in symbol table. BC contains symbol. Returns with HL pointing to symbol value.
CY = 1 => symbol was found.
CY = 0 and a scalar => symbol not found, but inserted and initialized to 0.
CY = 0 and an array => not found, no action taken: HL are meaningless.

LADD — Floating-point add routine.

LSUB — Floating-point subtract routine.

LOIU — Floating-point divide routine.

LMUL — Floating-point multiply routine.

LMCM — One of the floating-point routines. Compares two floating-point values, HL Point to first floating-point values and HB point to second floating-point value.
z = 1 = >Equality
Cy = 1 = >first<second
(Note: compares absolute only, does not reference mantissa sign.)

MCHK — Waits for flag from port 3. Proper mask is sent in register B.

MEMFUL — Checks to see if memory is full. HL point to location of memory to be checked. Memory is considered full if it is within 50_{10} locations of the current value of stack pointer.

MULT — Multiplies two two-byte binary numbers. HL point to last byte of four bytes. First two contain first number. Last two contain second number. Answer returns to BCDE.

NSRCH — Routine to locate source line in memory passed binary value of line number in DE. Returns address of line in HL, CY = 1 = > not found.

OUTR — Used by CONV (CURT) to pad output to output buffer.

PAD — Pads characters to output buffer. A contains character; B contains number of pads.

SYMSRT — Checks a character string to see if it is a BASIC symbol. HL contains address pointing to 1st character of symbol, C contains length of line that contains symbol. A contains type of symbol sought.
0 = command 1 = keyword
z = operator or delimiter 3 = function
Returns with 377_8 in a register if nothing found. Otherwise A contains symbol number in appropriate KDAT table. Thus, for symbol type 2, if a 4 is returned, the symbol found was the fourth one (starting with 0) in table KDAT3 (KDAT concatenated with 2 and 1 or A). CIS is updated, but HL is not.

TTYIW — Inputs a line from teletype. Stores starting address at location pointed to by HL. Line edits. Returns length of line in A register (maximum line length is 72 characters).

VALUE — Called with HL pointing to A variable, constant, or function; C contains line length, returns with DE pointing to floating-point value. HL, C are updated.

VAR — Called with HL pointing to character string, C has line length. Determines if character string is a variable. If so, returns with CY = 1, DE pointing to value (subscripts of arrays are evaluated, etc.). HL, C updated. If not, a variable returns CY = 0, HL, C untouched.

WRIT — Dumps contents of output buffer to teletype. Uses entry WRIT1 with D register equal to one to suppress CR/LF.

ZROL — Part of floating-point subroutines. Writes a floating-point zero, starting at location pointed to by HL.

The preceding list contains those subroutines most likely to be used by someone modifying BASIC. If you plan on using one of the routines, you should examine it and its comments carefully.

Variables—Following is a list of interpreter variables, with a description of each variable:

MEMST — Assembly time variable. Contains the first available RAM Location. This is where active variables start.

MEMEND — Asembly time variable. Contains the last available location in RAM.

SEND — Has value 6, used with RST instruction to print characters via ODT.

OBUFF — Output buffer, the first location contains the number of characters in the buffer + 1.

IBUF — Input buffer, occupies same area as OBUFF.

STLINE — Points to first source line to be executed. If no source, contains 177777_8.

NLINE, NL2, NL4, NL6 — Contain address, binary-equivalent line number, forward pointer, and length of next input line.

KLINE, KL2, KL4, KL6 — Same as above, but used by a subroutine that inserts lines in sequential order (insert).

PLINE, PL2, PL4, PL6 — Subroutine insert to order statements sequentially.

KASE, LEN — Temporary storage for command mode routines.

MULT1, MULT2 — Used to store binary values to be multiplied.

SBSAV — Temporary storage for call-statement processor.

STSPAC — Next available location in memory, symbol table starts here at run time.

LPNT — Pointer to the current line at run time.

CPNT — Pointer to current character in current line at run time.

KFPNT — Point to next sequential line at run time.

FREG1, FREG2 — Two floating-point registers.

HLINP, CREG — Temporary storage for HL and C registers for routine INP

NXTSP — Pointer to next available space of memory for symbol table.

GREG — General register, in and out instructions are stored here and executed for get and put functions.

MODE — Indicates to INP routine whether input data comes from source or teletype.

MESCR — Temporary storage for call-statement processor. Points to next available space after symbol table. Area after the symbol table is used to store intermediate results of expressions or constants passed to user subroutines.

VARAD — Temporary storage space for input-statement processor.

VEND — Assembly time variable. Indicates end of interpreter variable-storage area and where FWAM pointer is to go.

FWAM — First word of available memory pointer. This is where user source programs go.

Some of the above variables occupy the same area of memory. This is because some variables are used only in the command mode and others only at runtime. To conserve space, they share the same memory locations.

New BASIC Statements—To add additional statements to the BASIC, use the following procedure. First, insert the statement keyword in the data tables for subroutine SYMSRT. Then, insert the starting address of the statement processor in the interpreter JUMP table. Finally, the statement processor itself must be inserted.

The keyword must be entered in the table KDAT2. The first byte must be the keyword length and the next bytes hold the ASCII-coded keyword. The table must end with A 377_8. If the keyword is the Nth entry in the table, on return from SYMSRT, the A register will hold N-1 if the keyword is found.

The starting address of the statement processor must be inserted into table JTBL. The order of keywords in KDAT2 must correspond with statement processor addresses in JTBL since, on return from SYMSRT, the A register times two is used as offset in JTBL to determine processor address.

The statement processor must be placed somewhere in memory. Generally, the first thing done in the statement processors is to load the pointer to the statement (LHLD CPNT) and increment past the keyword (since HL is not updated by SYMSRT). On entry, C contains the number of characters in the line minus those checked by SYMSRT. The end of the processor should be a "JMPIED" instruction.

New Functions—New functions must be added to SYSSRT Data Table KDAT4 in the same manner as for key words. The function itself must be placed in subroutine "VALUE." Presently, the only function in VALUE is GET.

Message Lines—The following description tells how to incorporate messages into BASIC output routines. Currently, to output a message to the teletype, the user executes an LXI H,ODATA, then a call to FORMK where K is an integer indicating which message is wanted (i.e., K = z indicates "TURN ON PUNCH") FORM pads the message into the output buffer. Then A "CALL WRIT" writes the contents of the buffer.

Suppose the message "POTATO BASIC" is to be added. Preceding the form 9 instruction, we will insert "FOR10: INR L." At the end of the ODATA table, we add "DB ODAT8 and 377Q." And, after message ODAT 7, we add ODAT8 DB * 12, "POTATO BASIC." Now, the following program segment:

```
LXI     H,ODATA
CALL    FOR10
CALL    WRIT,
```

will cause "POTATO BASIC" to be output to the teletype.

ODT-80 is an octal debugging routine for use on the Intel 8080 microprocessor. This routine provides the capability to examine and modify all of the memory that is available to the micro-computer and transfer program control to the created program. ODT-80 makes use of simple keyboard commands from any terminal—such as a teletypewriter—that is attached to the system.

ODT-80 (Octal Debugging Technique) is a program written for the MCS-80 that allows the user to modify a program via a teletypewriter keyboard. The program occupies 400 octal words and must be located in the lowest memory page of the MCS-80 system, since the program uses the RESTART instructions.

ODT has been proved to be an effective aid to debugging on microprocessors. The first version,[1] for the Intel 8080,* has been "front panel" for virtually hundreds of microprocessor applications. The author submits this ODT for the 8080 in hopes that the tradition of soft panels may be perpetuated.

SYSTEM REQUIREMENTS

All addresses of memory locations and contents of memory locations are referred to in octal numbers. A question mark (?) will be typed for any illegal input.

The minimum system requirements for using ODT are as follows:

- MCS—80 computer set
- ODT programmable read only memory (PROM) at memory page 000_8
- 256 word (RAM) at page 010_8
- Teletype interface with the following codes:

  OUT 2—SEND ASCII character

  IN 2—Input word from TTY

  IN 3—Read Flags $\underset{D_7}{1}\ \underset{}{1}\underset{}{1}\ \underset{}{1}\underset{}{1}\underset{D_0}{1}$ (flag word)

  Sending Done

  Word Received

UTILITY ROUTINES

The following subroutines are available to the user as utility routines for other programs:

Address	Call	Routine
307	RST ODT	Restart ODT Program. This is useful for error branching in program debugging.
367	RST SEND	Send the ASCII character presently in the "A" register.

315	CAL READ	Wait for a character to be received from the teletype and return with the ASCII character in
333		the "A" register. The "A"
Ø		and "B" registers are used in this routine.
315	CALL CRLF	Send a carriage return and
370		a line feed to the teletypewriter.
Ø		The "A" register is used in this routine.
315	CAL OCTALP	Send a space and type in
301		Octal the three digit number in the "A" register.
Ø		
		The A, B, and E registers are used in this routine.

COMMANDS

n_8/) —The ASCII Slash (/) character is used to Open the n_8 address and type the contents in octal.

(LF) —The ASCII Line Feed (LF) character is used to close the currently open address, and open the next sequential address. The contents of the open register may be changed by typing the octal number to be input and then typing a (CR).

(.) —The ASCII Period (.) operator character may be used before the (/) operator to open the address last used.

(n_8S) —The SET command is used to set the H register to the memory page to be accessed.
(EX) 1ØS ;H = Ø1Ø$_8$

(n_8R) —The READ operator is used to start a loader program in memory page No. 2, location Ø.

If this command is preceded by an "nS" command, the n value will be passed to the loader program in the H register.

(n_8G) —The GO routine is used to start a program in memory. The octal number typed before the "G" operator will set the starting address in memory. The "G" operator should be preceded by an "nS" command to select the desired page.

(CTRL-C) —The ASCII Control C character is used to command the type out of the top two locations in the stack. (See trap.) Leaves the stack pointer at its initial position -2.

(CR) —The ASCII Carriage Return (CR) character is used to close the currently open address.

TRAP

The trap feature of ODT works as follows: When the central processor encounters a 377_8 as an instruction, the processor decodes this as a restart to location 70_8 in memory 0_8. An example of this is when a nonexistent memory is addressed, or when a 377_8 is placed in a program for a break point. At this time, ODT stores in the pushdown stack the address of the errant location and all of the registers. The trap indication is a "T" typed out on the teletypewriter. Control is now back in ODT and multiple Control-C's (CTRL-C) typed by the operator will yield, in sets of two (One set for each CTRL-C Typed), the trapped information in the following format:

ADDRESS ERROR + 1		REGISTERS							
PAGE NO.	LOCATION	A	F	B	C	D	E	H	L
XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX

The "F" register is the status flags of the 8080, with the format shown in Fig. 1.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
MSB (SIGN)	ZERO FLAG	Ø	CARRY 1	Ø	EVEN PARITY	1	CARRY 2

Figure 1. "F" register format.

A CRTL-C typed at any time will type out the pushdown stack but *not* in the format shown above! The above format is only available immediately after a "T" has been sent by ODT.

If a continuous string of "T" are sent to the TTY without stopping, this is an indication that the stack pointer is pointing at nonexistent memory. It will be necessary to restart ODT.

RESTART/INTERRUPT LOCATIONS

There are five segments of memory reserved in ODT for INTERRUPT or RESTART operation. These locations, shown in Fig. 2, contain jumps to the scratch RAM memory #1Ø, used by ODT. This allows in-

terrupt service to be handled even though memory #Ø is preprogrammed to contain ODT.

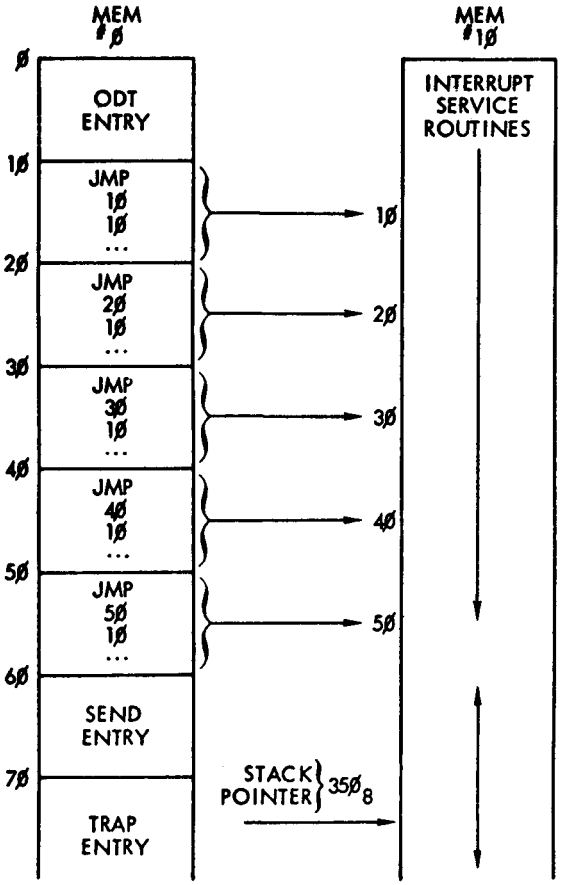


Figure 2. RESTART/INTERRUPT locations.

STACK POINTER

The stack pointer is reset to location $35Ø_8$ every time ODT is restarted via location Ø of ROM Ø—in other words, whenever a "?" is sent by ODT. ODT and other programs use the stack pointer; therefore, memory locations plus and minus this location are apt to be overwritten. However, the stack pointer moves down in memory (high addresses to low addresses), so normal operation will keep the stack pointer in RAM $1Ø_8$.

EXAMPLE OF TYPICAL DEBUGGING OPERATION

The following example illustrates a typical debugging operation. It is assumed that a program has been assembled and that the program is to be loaded by a loader in PROM 2.

1. Set the memory field with the S command, and read the tape with the R command.

```
1ØS*        ;SET MEMORY FIELD TO 1Ø
ØR          ;START READER ON TTY
?           ;WHEN TAPE IS READ IN ODT
            RESTARTS (startup of ODT
            depends upon the reader pro-
            gram in PROM 2)
```

2. Start the program with the G command.

```
1ØS         ;START PROGRAM AT LOCA-
            TION Ø
ØG
TEST PROGRAM ;TYPED BY PROGRAM WITH AN ERROR
```

3. Change location (30) to correct output. Restart ODT.

```
?
1ØS
30/327 322)   ;CHANGE "W" TO AN "R"

TEST PROGRAM ;CORRECTED OUTPUT
```

EXAMPLE PROGRAM LISTING

```
                              ;ODT TEST PROGRAM
                              ;10-21-74
000006              SEND EQU 6
000370              CRLF EQU 370Q
                              ;
                              ;PROGRAM PROPER
                              ;
004000                        ORG 4000Q
004000  041 022 010           LXI H TABLE          ;SET UP ADDRESS
004003  176           LOOP:   MOV A,M              ;GET ENTRY
004004  376 000               CPI 0
004006  312 016 010           JZ ND                ;IF 0 END OF LIST
004011  367                   RST SEND             ;TYPE CHARACTER
004012  043                   INX H
004013  303 003 010           JMP LOOP             ;LOOP


004016  315 370 000 ND:       CALL CRLF            ;SEND CRLF

004021  166                   HLT
                              ;
004022  324           TABLE:  DB 324Q                ;T
004023  305                   DB 305Q                ;E
004024  323                   DB 323Q                ;S
004025  324                   DB 324Q                ;T
004026  340                   DB 240Q                ;SPACE
```

```
004027   240              DB 320Q              ;P
004030   327              DB 327Q              ;W FOR ERROR
004031   317              DB 317Q              ;O
004032   307              DB 307Q              ;G
004033   322              DB 322Q              ;R
004034   301              DB 301Q              ;A
004035   315              DB 315Q              ;M
004036     Ø              DB Ø
                          END

NO PROGRAM ERRORS


                    SYMBOL TABLE

* 01

A       000007   B       000000   C       000001   CRLF    000370
D       000002   E       000003   H       000004   L       000005
LOOP    004003   M       000006   ND      004015   PSW     000006
SEND    000006   SP      000006   TABLE   004021
```

ODT-80 PROGRAM LISTING

```
SAMPLE JUMP TABLE *
SEND    EQU   344Q
READ    EQU   333Q
MCHK    EQU   113144Q

        ORG   400Q

SPNT:   DW    37777Q
SUBAD:  DW    SUBS
CONIN:  JMP   READ
CONOUT: JMP   SEND
STATUS: JMP   STAT
HSRDR:  JMP   HSR

        ORG   500Q

STAT:   IN    3              ; READ STATUS PORT
        RET

HSR:    MVI   B,4
        OUT   5
        CALL  MCHK
        IN    5
        RET

SUBS:   DB    377Q

        END
```

THE 6K 'MASK-ROM' VERSION OF THE 'LLL INTERPRETIVE
BASIC' OCCUPIES PAGES 200 THRU 227 (OCTAL).

ALL I/O IS DONE THRU A JUMP-TABLE LOCATED ON PAGE 1.
THIS JUMP-TABLE MUST BE LOCATED AT THE BELOW
SPECIFIED LOCATIONS (GIVEN IN OCTAL):

```
PAGE 1    LOC.  0     DW   ADDR.     ; ADDR. IS THE LOCATION
                                     ; OF THE STACK-POINTER

          LOC.  2     DW   ADDR.     ; ADDR. IS THE LOCATION OF
                                     ; THE USER-SUB TABLE
                                     ; (SEE UCID-17090 PAGE 9)

          LOC.  4     JMP  CONIN     ; CONIN IS LOCATION OF CONSOLE
                                     ; INPUT ROUTINE (BASIC EXPECTS
                                     ; CHAR. TO BE RETURNED IN THE
                                     ; B-REGISTER)

          LOC.  7     JMP  CONOUT    ; CONOUT IS LOCATION OF CONSOLE
                                     ; OUTPUT ROUTINE (BASIC PASSES
                                     ; CHAR. IN A-REGISTER)

          LOC. 12     JMP  STATUS    ; STATUS READS THE I/O STATUS
                                     ; PORT AND RETURNS THE FOLLOWING
                                     ; IN THE A-REGISTER TO BASIC -
                                     ;
                                     ; B7 B6 B5 B4 B3 B2 B1 B0
                                     ;
                                     ; B0=1   IF CHAR. AVAILABLE
                                     ;        FOR CONSOLE INPUT.
                                     ; B1=1   IF DATA TRANSMISSION
                                     ;        COMPLETE.
```

```
                LOC. 15     JMP   HSRDR     ; HSRDR IS LOCATION OF HIGH-
                                            ; SPEED PAPER-TAPE READER
                                            ; ROUTINE
                                            ; (BASIC EXPECTS CHAR. TO BE
                                            ;  RETURNED IN A-REGISTER)
```

BASIC INTERPRETER

```
0400                MEMST     EQU     2000Q           ;MUST BE ON PAGE BOUNDARY
                    ; DEFINE I-O/SP/SUB ADD. JUMP TABLE LOCATIONS
0100                SPNT      EQU     400Q
0102                SUBAD     EQU     402Q
0104                CONIN     EQU     404Q
0107                CONOUT    EQU     407Q
010A                STATUS    EQU     412Q
010D                HSRDR     EQU     415Q
                    ;
0400                OBUFF     EQU     MEMST           ;INPUT AND OUTPUT BUFFERS OCCUPY
0401                IBUF      EQU     MEMST+1         ;SAME AREA
0449                STLINE    EQU     MEMST+111Q
044B                NLINE     EQU     MEMST+113Q
044D                NL2       EQU     MEMST+115Q
044F                NL4       EQU     MEMST+117Q
0451                NL6       EQU     MEMST+121Q
0452                KLINE     EQU     MEMST+122Q
0454                KL2       EQU     MEMST+124Q
0456                KL4       EQU     MEMST+126Q
0458                KL6       EQU     MEMST+130Q
0459                PLINE     EQU     MEMST+131Q
045B                PL2       EQU     MEMST+133Q
045D                PL4       EQU     MEMST+135Q
045D                SBSAV     EQU     PL4             ;RETURN ADD. SAVE FOR CALL STMT.
045F                PL6       EQU     MEMST+137Q
0460                KASE      EQU     MEMST+140Q
0461                LEN       EQU     MEMST+141Q
0462                MULT1     EQU     MEMST+142Q
0464                MULT2     EQU     MEMST+144Q
0459                NXTSP     EQU     MEMST+131Q
044B                STSPAC    EQU     MEMST+113Q
```

```
97C7                    NORM    EQU     113707Q
97CA                    FLOAT   EQU     113712Q
97CD                    ZROL    EQU     113715Q
0452                    LPNT    EQU     MEMST+122Q
0458                    KLEN    EQU     MEMST+130Q
045B                    CPNT    EQU     MEMST+133Q
0456                    KFPNT   EQU     MEMST+126Q
0480                    FREG2   EQU     MEMST+200Q
0484                    CREG    EQU     MEMST+204Q
97D0                    LADD    EQU     113720Q
97D3                    LMUL    EQU     113723Q
97D6                    LDIV    EQU     113726Q
97D9                    LSUB    EQU     113731Q
97DC                    DFXL    EQU     113734Q
97DF                    LMCM    EQU     113737Q
0486                    HLINP   EQU     MEMST+206Q
0477                    GREG    EQU     MEMST+167Q
047C                    FREG1   EQU     MEMST+174Q
0466                    SCR     EQU     MEMST+146Q
97E5                    CONV    EQU     113745Q
0485                    MODE    EQU     MEMST+205Q
97E8                    FINPT   EQU     113750Q
97EB                    MULT    EQU     113753Q
97EE                    PTVAL   EQU     113756Q
97F1                    DCOMP   EQU     113761Q
97F4                    MCHK    EQU     113764Q
97F7                    CHAR2   EQU     113767Q
0488                    MESCR   EQU     MEMST+210Q      ;DEFINE MEMORY SCR AREA PNTR
048A                    VARAD   EQU     MEMST+212Q      ;TEMP SPACE FOR INP. STMT.
048C                    VNAME   EQU     MEMST+214Q      ;TEMP SPACE FOR 'FOR-NEXT'
048E                    VLOC    EQU     MEMST+216Q      ;TEMP SPACE FOR 'FOR-NEXT'
0490                    FLIMT   EQU     MEMST+220Q      ;TEMP SPACE FOR 'FOR-NEXT'
0494                    NEST    EQU     MEMST+224Q      ;NESTING STACK-POINTER
0496                    STAC    EQU     MEMST+226Q      ;FOR-NEXT NESTING STACK
0014                    STSIZ   SET     20              ;STACK SIZE. ALLOWS 10 NESTED
0496                    TOPNS   EQU     STAC            ;FOR-NEXT TOP OF STACK
04AA                    BOTNS   EQU     STAC+STSIZ      ;BOTTOM OF STACK
04AA                    VEND    EQU     MEMST+252Q      ;DEF. END OF VAR. STORAGE AREA
                        ;  MAIN ROUTINE--HANDLES ALL USER INPUT
8000                            ORG     100000Q
8000    21 00 04        M1:     LXI     H,0BUFF
```

```
8003    36 01              MVI     M,1
8005    21 49 04           LXI     H,STLINE
8008    36 FF              MVI     M,377Q
800A    2C                 INR     L
800B    36 FF              MVI     M,377Q
800D    2A DA 8F           LHLD    FWAM            ;GET ADDRES OF FWA MEM.
8010    22 4B 04           SHLD    NLINE           ;STORE IN FREE SPACE PNTR.
8013    2A 00 01    M1A:   LHLD    SPNT
8016    F9                 SPHL
8017    21 69 84    M2:    LXI     H,ODATA
801A    CD 57 84           CALL    FORM1
801D    CD D3 82           CALL    WRIT
8020    2A 4B 04    M3:    LHLD    NLINE
8023    23                 INX     H
8024    23                 INX     H
8025    23                 INX     H
8026    23                 INX     H
8027    23                 INX     H
8028    CD D2 83           CALL    TTYIN
802B    4F                 MOV     C,A
802C    FE 00              CPI     0
802E    CA 20 80           JZ      M3
8031    CD 25 82           CALL    ALPHA
8034    DA 43 80           JC      M4
8037    CD 17 82           CALL    NUMB
803A    D4 9F 81           CNC     WHAT
803D    CD 90 80           CALL    INSERT
8040    C3 20 80           JMP     M3
8043    3E 00       M4:    MVI     A,0
8045    CD 00 83           CALL    SYMSRT
8048    3C          M4A:   INR     A
8049    CC 9F 81           CZ      WHAT
804C    3D                 DCR     A
804D    CA 85 86           JZ      RUN
8050    3D                 DCR     A
8051    CC AB 81           CZ      TAPE
8054    CA 17 80           JZ      M2
8057    3D                 DCR     A
8058    CC D2 81           CZ      LIST
805B    CA 17 80           JZ      M2
805E    3D                 DCR     A
```

```
805F CA 00 80          JZ      M1
8062 3D                DCR     A
8063 C4 9F 81          CNZ     WHAT
                ; ROUTINE TO INPUT FROM HSR
8066 CD 17 85   PTAPE: CALL    CHAR5
8069 FE 00             CPI     0
806B CA 66 80          JZ      PTAPE
806E CD E4 84   PT1:   CALL    HSRIN
8071 4F                MOV     C,A .
8072 FE 00             CPI     0
8074 CA 66 80          JZ      PTAPE
8077 CD 25 82          CALL    ALPHA
807A DA 43 80          JC      M4
807D CD 90 80          CALL    INSERT
8080 CD 17 85          CALL    CHAR5
8083 FE 00             CPI     0
8085 CA 17 80          JZ      M2
8088 23                INX     H
8089 23                INX     H
808A 23                INX     H
808B 23                INX     H
808C 23                INX     H
808D C3 6E 80          JMP     PT1
                ; ROUTINE TO HANDLE ALL SOURCE LINE INPUT.
                ; THIS INCLUDES INSERTION, DELEATION, AND
                ; ADDITION OF NEW SOURCE LINES.
8090 2B         INSERT: DCX     H
8091 71                MOV     M,C
8092 23                INX     H
8093 CD 2D 82          CALL    CVB
8096 FE 05             CPI     5
8098 DA A3 80          JC      ISR1A
809B C4 9F 81          CNZ     WHAT
809E 7B                MOV     A,E
809F 17                RAL
80A0 DC 9F 81          CC      WHAT
80A3 2A 4B 04   ISR1A: LHLD    NLINE
80A6 72                MOV     M,D
80A7 23                INX     H
80A8 73                MOV     M,E
80A9 21 4B 04          LXI     H,NLINE
```

```
80AC CD EE 97          CALL  PTVAL
80AF 2A 49 04          LHLD  STLINE
80B2 CD A6 82          CALL  CHK1
80B5 D2 80 04          JNC   ISRT3
80B8 2A 4B 04          LHLD  NLINE
80BB 22 49 04          SHLD  STLINE
80BE 16 FF             MVI   D,377Q
80C0 5A                MOV   E,D
80C1 CD 8C 81          CALL  STPNT
80C4 23                INX   H
80C5 7E          ISRT2: MOV   A,M
80C6 C6 05             ADI   5
80C8 2A 4B 04          LHLD  NLINE
80CB 85                ADD   L
80CC 6F                MOV   L,A
80CD 3E 00             MVI   A,0
80CF 8C                ADC   H
80D0 67                MOV   H,A
80D1 22 4B 04          SHLD  NLINE
80D4 C9                RET
80D5 22 52 04    ISRT3: SHLD  KLINE
80D8 21 52 04    ISRT4: LXI   H,KLINE
80DB CD EE 97          CALL  PTVAL
80DE 21 4D 04          LXI   H,NL2
80E1 56                MOV   D,M
80E2 2C                INR   L
80E3 5E                MOV   E,M
80E4 21 54 04          LXI   H,KL2
80E7 46                MOV   B,M
80E8 2C                INR   L
80E9 4E                MOV   C,M
80EA CD F1 97          CALL  DCOMP
80ED CA 22 81          JZ    ISRT6
80F0 DA 7B 81          JC    ISR12
80F3 2A 56 04          LHLD  KL4
80F6 CD A6 82          CALL  CHK1
80F9 DA 10 81          JC    ISRT5
80FC E5                PUSH  H
80FD 2A 52 04          LHLD  KLINE
8100 22 59 04          SHLD  PLINE
8103 21 59 04          LXI   H,PLINE
```

```
8106    CD EE 97                CALL    PTVAL
8109    E1                      POP     H
810A    22 52 04                SHLD    KLINE
810D    C3 D8 80                JMP     ISRT4
8110    2A 4B 04        ISRT5:  LHLD    NLINE
8113    CD 92 81                CALL    NOLINE
8116    C8                      RZ
8117    EB                      XCHG
8118    2A 52 04                LHLD    KLINE
811B    CD 8C 81                CALL    STPNT
811E    EB                      XCHG
811F    C3 BE 80                JMP     ISRT1
8122    2A 4B 04        ISRT6:  LHLD    NLINE
8125    CD 92 81                CALL    NOLINE
8128    C2 49 81                JNZ     ISRT8
812B    2A 49 04                LHLD    STLINE
812E    EB                      XCHG
812F    2A 52 04                LHLD    KLINE
8132    E5                      PUSH    H
8133    C1                      POP     B
8134    CD F1 97                CALL    DCOMP
8137    2A 56 04                LHLD    KL4
813A    CA 45 81                JZ      ISRT7
813D    EB                      XCHG
813E    2A 59 04                LHLD    PLINE
8141    CD 8C 81                CALL    STPNT
8144    C9                      RET
8145    22 49 04        ISRT7:  SHLD    STLINE
8148    C9                      RET
8149    2A 56 04        ISRT8:  LHLD    KL4
814C    EB                      XCHG
814D    2A 4B 04                LHLD    NLINE
8150    CD 8C 81                CALL    STPNT
8153    2A 52 04        ISRT9:  LHLD    KLINE
8156    EB                      XCHG
8157    2A 49 04                LHLD    STLINE
815A    E5                      PUSH    H
815B    C1                      POP     B
815C    CD F1 97                CALL    DCOMP
815F    CA 72 81                JZ      ISR11
8162    2A 4B 04                LHLD    NLINE
```

```
8165  EB                        XCHG
8166  2A 59 04                  LHLD   PLINE
8169  CD 8C 81                  CALL   STPNT
816C  21 51 04          ISR10:  LXI    H,NL6
816F  C3 C5 80                  JMP    ISRT2
8172  2A 4B 04          ISR11:  LHLD   NLINE
8175  22 49 04                  SHLD   STLINE
8178  C3 6C 81                  JMP    ISR10
817B  2A 52 04          ISR12:  LHLD   KLINE
817E  EB                        XCHG
817F  2A 4B 04                  LHLD   NLINE
8182  CD 92 81                  CALL   NOLINE
8185  C8                        RZ
8186  CD 8C 81                  CALL   STPNT
8189  C3 53 81                  JMP    ISRT9
                        ; ROUTINE TO STORE POINTERS INTO MEM ARRAY
818C  23                STPNT:  INX    H
818D  23                        INX    H
818E  73                        MOV    M,E
818F  23                        INX    H
8190  72                        MOV    M,D
8191  C9                        RET
                        ; ROUTINE TO CHECK NEW LINE FOR SOURCE STMT.
8192  E5                NOLINE: PUSH   H
8193  23                        INX    H
8194  23                        INX    H
8195  23                        INX    H
8196  23                        INX    H
8197  4E                        MOV    C,M
8198  23                        INX    H
8199  CD 74 82                  CALL   LENGTH
819C  E1                        POP    H
819D  B9                        CMP    C
819E  C9                        RET
                        ; ROUTINE TO RESPOND WITH 'WHAT?' FOR UNIDENTIFIED
                        ; COMMAND.
819F  21 69 84          WHAT:   LXI    H,ODATA
81A2  CD 53 84                  CALL   FORM7
81A5  CD D3 82                  CALL   WRIT
81A8  C3 13 80                  JMP    M1A
                        ; ROUTINE TO PUNCH PAPER TAPE OF SOURCE.
```

```
81AB  F5           TAPE:   PUSH   PSW
81AC  C5                   PUSH   B
81AD  21 69 84             LXI    H,ØDATA
81B0  CD 56 84             CALL   FØRM2
81B3  CD D3 82             CALL   WRIT
81B6  3E 00                MVI    A,0
81B8  C1                   PØP    B
81B9  06 40                MVI    B,100Q
81BB  F5                   PUSH   PSW
81BC  C5                   PUSH   B
81BD  CD B0 82             CALL   PAD
81C0  CD D3 82             CALL   WRIT
81C3  C1                   PØP    B
81C4  C5                   PUSH   B
81C5  CD D2 81             CALL   LIST
81C8  C1                   PØP    B
81C9  F1                   PØP    PSW
81CA  CD B0 82             CALL   PAD
81CD  CD D3 82             CALL   WRIT
81D0  F1                   PØP    PSW
81D1  C9                   RET
                    ; RØUTINE TØ LIST TØ TTY THE SØURCE STMTS.
81D2  2A 49 04      LIST:  LHLD   STLINE
81D5  CD A6 82             CALL   CHK1
81D8  DA 13 80             JC     M1A
81DB  22 59 04             SHLD   PLINE
81DE  21 FF FF             LXI    H,177777Q
81E1  22 52 04             SHLD   KLINE
81E4  0D                   DCR    C
81E5  C4 57 85             CNZ    BØUND
81E8  2A 59 04             LHLD   PLINE
81EB  23            LIS1:  INX    H
81EC  23                   INX    H
81ED  46                   MØV    B,M
81EE  23                   INX    H
81EF  4E                   MØV    C,M
81F0  C5                   PUSH   B
81F1  23                   INX    H
81F2  CD 58 84             CALL   FØRM5
81F5  CD D3 82             CALL   WRIT
81F8  C1                   PØP    B
```

```
81F9  2A 52 04            LHLD   KLINE
81FC  EB                  XCHG
81FD  CD F1 97            CALL   DCOMP
8200  C8                  RZ
8201  68                  MOV    L,B
8202  61                  MOV    H,C
8203  CD 09 82            CALL   QUITT              ;CHECK FOR INTERRUPTION
8206  C3 EB 81            JMP    LIS1               ;NONE - CONTINUE
                    ;THIS ROUTINE CHECKS PORT 2 FOR A CNTRL/S CHARACTER
                    ;IF ONE IS FOUND THEN EXECUTION IS TO BE INTERRUPTED
                    ;CONTROL IS PASSED TO M1A
8209  CD 0A 01      QUITT: CALL   STATUS             ;TEST FLAG PORT
820C  1F                  RAR                       ;FLAG TO CY
820D  D0                  RNC                       ;NOTHING THERE
820E  CD 04 01            CALL   CONIN              ;FLAG WAS SET, GET DATA
8211  FE 93          QTCHK: CPI    223Q              ;WAS IT CNTRL/S?
8213  CA 13 80            JZ     M1A                ;YES
8216  C9                  RET                       ;NO, RETURN
                    ;  ROUTINES NUMB AND ALPHA CHECK IF CONTENTS OF MEMORY
                    ;  LOCATION IN HL CONTAIN ASCII NUMERIC OR ALPHBETIC
                    ; CHARACTER. RETURN CY=1 IF YES, CY=0 IF NO.
8217  C5            NUMB:  PUSH   B
8218  06 B0                MVI    B,260Q
821A  0E BA                MVI    C,272Q
821C  7E              C1:  MOV    A,M
821D  B8                   CMP    B
821E  3F                   CMC
821F  D2 23 82             JNC    BAC
8222  B9                   CMP    C
8223  C1            BAC:   POP    B
8224  C9                   RET
8225  C5            ALPHA: PUSH   B
8226  06 C1                MVI    B,301Q
8228  0E DB                MVI    C,333Q
822A  C3 1C 82             JMP    C1
                    ; ROUTINE TO CONVERT ASCII NUMERIC CHAR. STRING TO
                    ; EQUIVALENT BINARY NUMBER. RETURNS EQUIVALENT IN
                    ; DE REG.  LENGTH OF LINE PASSED IN REG C AND
                    ; RETURNED POINTING TO LAST NUMERIC CHAR. LENGTH
                    ; OF CHAR STRING RETURNED IN REG A.
822D  E5            CVB:   PUSH   H
```

```
822E C5          PUSH  B
822F CD 74 82    CALL  LENGTH
8232 F5          PUSH  PSW
8233 E5          PUSH  H
8234 FE 00       CPI   0
8236 CA 6B 82    JZ    CVB2
8239 21 60 04    LXI   H,KASE
823C 77          MOV   M,A
823D 2C          INR   L
823E 71          MOV   M,C
823F 21 0A 00    LXI   H,10
8242 22 62 04    SHLD  MULT1
8245 21 00 00    LXI   H,0
8248 22 64 04    SHLD  MULT2
824B 21 65 04    LXI   H,MULT2+1
824E CD EB 97    CALL  MULT
8251 E3          XTHL
8252 7E          MOV   A,M
8253 D6 30       SBI   '0'
8255 82          ADD   D
8256 57          MOV   D,A
8257 3E 00       MVI   A,0
8259 8B          ADC   E
825A 5F          MOV   E,A
825B 23          INX   H
825C E3          XTHL
825D 72          MOV   M,D
825E 2C          INR   L
825F 73          MOV   M,E
8260 E5          PUSH  H
8261 21 61 04    LXI   H,LEN
8264 35          DCR   M
8265 2D          DCR   L
8266 35          DCR   M
8267 E1          POP   H
8268 C2 4E 82    JNZ   CVB1
826B E1          POP   H
826C F1          POP   PSW
826D C1          POP   B
826E 21 61 04    LXI   H,LEN
8271 4E          MOV   C,M
CVB1:
CVB2:
```

```
8272    E1                      POP     H
8273    C9                      RET
                        ; ROUTINE TO EVALUATE LENGTH OF ASCII NUMERIC
                        ; CHAR STRING: PASSED ADD OF FIRST CHAR IN HL REG.
                        ; RETURNS LENGTH IN REG A.
8274    C5              LENGTH: PUSH    B
8275    E5                      PUSH    H
8276    06 00                   MVI     B,0
8278    CD 17 82        NLE1:   CALL    NUMB
827B    D2 87 82                JNC     NLE2
827E    23                      INX     H
827F    04                      INR     B
8280    0D                      DCR     C
8281    CA 87 82                JZ      NLE2
8284    C3 78 82                JMP     NLE1
8287    78              NLE2:   MOV     A,B
8288    E1                      POP     H
8289    C1                      POP     B
828A    C9                      RET
                        ; ROUTINE TO LOCATE SOURCE LINE IN MEM. PASSED BIN VALUE
                        ;OF LINE NUMBER IN DE(LOW,HIGH) REG. RETURNS ADDRESS OF
                        ;SOURCE LINE IN HL REGS.(HIGH,LOW). CY SET=@ NOT FOUND.
828B    2A 49 04        NSRCH:  LHLD    STLINE
828E    CD A6 82        L2:     CALL    CHK1
8291    D8                      RC
8292    46                      MOV     B,M
8293    23                      INX     H
8294    4E                      MOV     C,M
8295    CD F1 97                CALL    DCOMP
8298    CA A3 82                JZ      FOUND
829B    23                      INX     H
829C    7E                      MOV     A,M
829D    23                      INX     H
829E    66                      MOV     H,M
829F    6F                      MOV     L,A
82A0    C3 8E 82                JMP     L2
82A3    2B              FOUND:  DCX     H
82A4    B7                      ORA     A
82A5    C9                      RET
                        ; ROUTINE TO COMPARE CONTENTS OF HL TO 177777Q.
                        ; RETURNS CY=1 IF YES: CY=0 IF NO.
```

```
82A6 C5          CHK1:   PUSH  B
82A7 E5                  PUSH  H
82A8 06 00               MVI   B,0
82AA 0E 01               MVI   C,1
82AC 09                  DAD   B
82AD E1                  POP   H
82AE C1                  POP   B
82AF C9                  RET
                 ; ROUTINE TO PAD OUTPUT BUFFER WITH CONTENTS OF REG A.
                 ;REG B CONTAINS NUMBER OF CHAR TO PAD.
82B0 C5          PAD:    PUSH  B
82B1 D5                  PUSH  D
82B2 E5                  PUSH  H
82B3 21 00 04            LXI   H,0BUFF
82B6 4D                  MOV   C,L
82B7 6E                  MOV   L,M
82B8 57                  MOV   D,A
82B9 3E 49               MVI   A,73
82BB BD          P1:     CMP   L
82BC C2 82               JNZ   P2
82BF 69                  MOV   L,C
82C0 77                  MOV   M,A
82C1 CD D3 82            CALL  WRIT
82C4 2C                  INR   L
82C5 72          P2:     MOV   M,D
82C6 2C                  INR   L
82C7 05                  DCR   B
82C8 C2 BB 82            JNZ   P1
82CB 7A                  MOV   A,D
82CC 45                  MOV   B,L
82CD 69                  MOV   L,C
82CE 70                  MOV   M,B
82CF E1                  POP   H
82D0 D1                  POP   D
82D1 C1                  POP   B
82D2 C9                  RET
                 ; ROUTINE TO DUMP OUTPUT BUFFER TO TTY.
82D3 16 00       WRIT:   MVI   D,0
82D5 F5          WRIT1:  PUSH  PSW
82D6 E5                  PUSH  H
82D7 C5                  PUSH  B
```

```
82D8  21 00 04            LXI   H,0BUFF
82DB  E5                  PUSH  H
82DC  4E                  MOV   C,M
82DD  0D                  DCR   C
82DE  CA F9 82            JZ    W2
82E1  2C                  INR   L
82E2  7E            W1:   MOV   A,M
82E3  CD 07 01            CALL  CONOUT          ;PRINT VIA ODT
82E6  2C                  INR   L
82E7  0D                  DCR   C
82E8  C2 E2 82            JNZ   W1
82EB  15                  DCR   D
82EC  CA F9 82            JZ    W2
82EF  3E 8D               MVI   A,215Q
82F1  CD 07 01            CALL  CONOUT          ;PRINT VIA ODT
82F4  3E 8A               MVI   A,212Q
82F6  CD 07 01            CALL  CONOUT          ;PRINT VIA ODT
82F9  E1            W2:   POP   H
82FA  36 01               MVI   M,1
82FC  C1                  POP   B
82FD  E1                  POP   H
82FE  F1                  POP   PSW
82FF  C9                  RET
                    ; ROUTINE TO LOCATE COMMANDS, KEY WORDS, OPERATORS,
                    ; AND FUNCTION.  HL CONTAINS ADD OF FIRST CHAR.:
                    ; REG C CONTAINS LENGTH OF LINE: RETURNS SYMBOL NUMBER
                    ; IF FOUND IN REG A, 377Q IN A IF NOT FOUND.
8300  D5            SYMSRT: PUSH  D
8301  C5                    PUSH  B
8302  E5                    PUSH  H
8303  E5                    PUSH  H
8304  21 61 04              LXI   H,LEN           ;SAVE C IN LEN
8307  71                    MOV   M,C
8308  21 54 83              LXI   H,KDATA         ;LOCATE TYPE OF SYMBOL SOUGHT.
830B  1E 00                 MVI   E,0             ;REG A CONTAINS:
830D  85                    ADD   L               ; 0 FOR COMMAND
830E  6F                    MOV   L,A             ; 1 FOR KEYWORD
830F  6E                    MOV   L,M             ; 2 FOR OPERATOR AND DELIMITER
8310  4E            S2:     MOV   C,M             ; 3 FOR FUNCTION
8311  2C            S3:     INR   L
8312  46                    MOV   B,M
```

```
8313 E3          XTHL
8314 7E          MOV   A,M
8315 B8          CMP   B
8316 C2 83       JNZ   S4
8319 0D          DCR   C
831A CA 48 83    JZ    S5
831D E5          PUSH  H
831E 21 61 04    LXI   H,LEN
8321 35          DCR   M
8322 E1          POP   H
8323 CA 2B 83    JZ    S4A
8326 23          INX   H
8327 E3          XTHL
8328 C3 11 83    JMP   S3
832B 0C          INR   C
832C E1          POP   H
832D 79          MOV   A,C
832E 85          ADD   L
832F 54          MOV   D,H
8330 E1          POP   H
8331 C1          POP   B
8332 C5          PUSH  B
8333 E5          PUSH  H
8334 21 61 04    LXI   H,LEN
8335 71          MOV   M,C
8338 6F          MOV   L,A
8339 62          MOV   H,D
833A 7E          MOV   A,M
833B 1C          INR   E
833C 4F          MOV   C,A
833D 3C          INR   A
833E C2 11 83    JNZ   S3
8342 21 61 04    LXI   H,LEN
8345 34          INR   M
8346 1E FF       MVI   E,377Q
8348 7B    S5:   MOV   A,E     ; MOVE SYMBOL NUMBER INTO REG A
8349 21 61 04    LXI   H,LEN
834C 5E          MOV   E,M
834D 1D          DCR   E
834E E1          POP   H
```

```
834F   E1                        POP     H
8350   C1                        POP     B
8351   4B                        MOV     C,E             ;MOVE NUMBER OF CHAR. LEFT IN
8352   D1                        POP     D               ;LINE INT
8353   C9                        RET
                    ;*************************************************
                    ;THE CODE FROM HERE TO THE NEXT LINE OF *'S MUST BE ON ONE PAGE
                    ;THIS MACRO ADDS PARITY BITS TO CHARACTERS
     1              CHPAR     MACRO   CH
     1                        DB      CH OR 200Q
                              ENDM
8354   58           KDATA:    DB      KDAT1 AND 377Q
8355   6D                     DB      KDAT2 AND 377Q
8356   A7                     DB      KDAT3 AND 377Q
8357   C9                     DB      KDAT4 AND 377Q
8358   03 D2 D5 CE  KDAT1:    DB      3,322Q,325Q,316Q  ;RUN
835C   03 D0 CC D3             DB      3,320Q,314Q,323Q  ;PLS
8360   03 CC C9 D3             DB      3,314Q,311Q,323Q  ;LIS
8364   03 D3 C3 D2             DB      3,323Q,303Q,322Q  ;SCR
8368   03 D0 D4 C1             DB      3,320Q,324Q,301Q  ;PTA
836C   FF                      DB      377Q
836D   03 CC C5 D4  KDAT2:    DB      3,314Q,305Q,324Q  ;LET
8371   03 D0 D2 C9             DB      3,320Q,322Q,311Q  ;PRI
8375   03 D2 C5 CD             DB      3,322Q,305Q,315Q  ;REM
8379   03 D3 D4 CF             DB      3,323Q,324Q,317Q  ;STO
837D   03 C5 CE C4             DB      3,305Q,316Q,304Q  ;END
8381   03 C7 CF D4             DB      3,307Q,317Q,324Q  ;GOT
8385   02 C9 C6                DB      2,311Q,306Q       ;IF
8388   03 C9 CE D0             DB      3,311Q,316Q,320Q  ;INP
838C   03 C4 C9 CD             DB      3,304Q,311Q,315Q  ;DIM
8390   03 C3                   DB      3,'C'+200Q        ;CAL
8392   C1                      DB      'A'+200Q
8393   CC                      DB      'L'+200Q
8394   04 C7                   DB      4,'G'+200Q        ;GOSU
8396   CF                      DB      'O'+200Q
8397   D3                      DB      'S'+200Q
8398   D5                      DB      'U'+200Q
8399   03 D2                   DB      3,'R'+200Q        ;RET
839B   C5                      DB      'E'+200Q
839C   D4                      DB      'T'+200Q
839D   03 C6                   DB      3,'F' OR 200Q     ;FOR
```

```
839F    CF                          DB      'O' OR 200Q
83A0    D2                          DB      'R' OR 200Q
83A1    04 CE                       DB      4,'N' OR 200Q       ;NEXT
83A3    C5                          DB      'E' OR 200Q
83A4    D8                          DB      'X' OR 200Q
83A5    D4                          DB      'T' OR 200Q
83A6    FF                          DB      377Q
                        ;DELIMITERS HAVE FOLLOWING VALUES:
                        ;
                        ;           <       0
                        ;           >       1
                        ;           ,       2
                        ;           =       3
                        ;           )       4
                        ;           ;       5
                        ;           THEN    6
                        ;           TO      7
                        ;           STEP    8
                        ;           *       9
                        ;           /       10
                        ;           +       11
                        ;           -       12
83A7    01 BC 01 BE     KDAT3:  DB      1,274Q,1,276Q       ;'<','>',
83AB    01 AC 01 BD             DB      1,254Q,1,275Q       ;',','=',
83AF    01 A9                   DB      1,251Q              ;')',
83B1    01 BB                   DB      1,';'+200Q          ;';'
83B3    04                      DB      4                   ;THEN
      1                 +       CHPAR   '''T'''
83B4  1 D4              +       DB      'T' OR 200Q
      1                 +       CHPAR   '''H'''
83B5  1 C8              +       DB      'H' OR 200Q
      1                 +       CHPAR   '''E'''
83B6  1 C5              +       DB      'E' OR 200Q
      1                 +       CHPAR   '''N'''
83B7  1 CE              +       DB      'N' OR 200Q
83B8    02                      DB      2                   ;TO
      1                 +       CHPAR   '''T'''
83B9  1 D4              +       DB      'T' OR 200Q
      1                 +       CHPAR   '''O'''
83BA  1 CF              +       DB      'O' OR 200Q
```

```
83BB    04                      +       DB      4               ;STEP
        1                       +       CHPAR   '''S'''
83BC    1 D3                    +       DB      'S' OR 200Q
        1                       +       CHPAR   '''T'''
83BD    1 D4                    +       DB      'T' OR 200Q
        1                       +       CHPAR   '''E'''
83BE    1 C5                    +       DB      'E' OR 200Q
        1                       +       CHPAR   '''P'''
83BF    1 D0                    +       DB      'P' OR 200Q
83C0    01 AA                           DB      1,'*'+200Q      ;'*'
83C2    01 AF 01 AB                     DB      1,257Q,1,253Q   ;'/','+'
83C6    01 AD                           DB      1,255Q          ;'-'
83C8    FF                              DB      377Q
83C9    03 C7 C5 D4     KDAT4:  DB      3,307Q,305Q,324Q ;GET
83CD    03 D0 D5 D4                     DB      3,320Q,325Q,324Q ;PUT
83D1    FF                              DB      377Q
                        ;*************************************************
                        ; ROUTINE TO INPUT SOURCE LINE FROM TTY. PASSED ADD
                        ; OF FIRST CHAR IN HL. RETURNS LENGTH OF LINE IN REG A
83D2    E5              TTYIN:  PUSH    H
83D3    06 00                   MVI     B,0
83D5    CD F7 97        TIN1:   CALL    CHAR2
83D8    FE 99                   CPI     231Q            ;CNTRL Y?
83DA    CA 0C 84                JZ      TIN5
83DD    FE FF                   CPI     377Q            ;RUBOUT?
83DF    CA FF 83                JZ      TIN2
83E2    FE DF                   CPI     337Q            ;BACK ARROW (RUBOUT)?
83E4    CA 02 84                JZ      TIN2+3
83E7    FE 8A                   CPI     212Q            ;LF?
83E9    CA D5 83                JZ      TIN1
83EC    FE 8D                   CPI     215Q            ;CR
83EE    CA 15 84                JZ      TIN4
83F1    FE 8C                   CPI     214Q            ;FORM FEED?
83F3    CA D5 83                JZ      TIN1            ;IGNORE
83F6    77                      MOV     M,A
83F7    23                      INX     H
83F8    04                      INR     B
83F9    CD 1D 85                CALL    MEMFUL
83FC    C3 D5 83                JMP     TIN1
83FF    3E DF           TIN2:   MVI     A,337Q
8401    CD 07 01                CALL    CONOUT          ;PRINT VIA ODT
```

```
8404  2B                        DCX   H
8405  05                        DCR   B
8406  F2 D5 83                  JP    TIN1
8409  E1                        POP   H
840A  AF                        XRA   A                 ;ZERO A
840B  C9                        RET
840C  3E DC             TIN5:   MVI   A,334Q
840E  CD 07 01                  CALL  CONOUT            ;PRINT VIA ODT
8411  3E 00             TIN5A:  MVI   A,0
8413  E1                        POP   H
8414  C9                        RET
8415  3E 8A             TIN4:   MVI   A,212Q
8417  CD 07 01                  CALL  CONOUT            ;PRINT VIA ODT
841A  0E 00             TIN4A:  MVI   C,0
841C  E1                        POP   H
841D  78                        MOV   A,B
841E  B9                        CMP   C
841F  C8                        RZ
                ;ROUTINE TO REMOVE BLANKS FROM SOURCE UNLESS ENCLOSED IN
8420  D5                       PUSH  D                 ;SAVE REG'S
8421  E5                       PUSH  H
8422  E5                       PUSH  H
8423  1E A2                    MVI   E,'"'+200Q        ;INIT E FOR COMPARES
8425  16 00                    MVI   D,0               ;D=1=>WITHIN QUOTES, LEAVE BLANKS
8427  AF              PK1:     XRA   A                 ;CLEAR A
8428  BA                       CMP   D                 ;CHECK INPUT MODE
8429  7E                       MOV   A,M               ;GET CHAR
842A  C2 3A 84                 JNZ   QSTRG             ;WITHIN QUOTE STRING
842D  BB                       CMP   E                 ;IS IT 1ST
842E  C2 35 84                 JNZ   $+7               ;NO - PROCEED
8431  14                       INR   D                 ;YES, SET FLAG
8432  C3 3F 84                 JMP   QSTR1             ;CONTINUE
8435  FE A0                    CPI   240Q              ;IS IT A SPACE?
8437  CA 44 84                 JZ    PK2               ;YES - LEAVE OUT
843A  BB              QSTRG:   CMP   E                 ;2ND "?
843B  C2 3F 84                 JNZ   $+4               ;NO - CONTINUE
843E  15                       DCR   D                 ;RESET FLAG
843F  E3              QSTR1:   XTHL                    ;GET DESTINATION ADDRESS
8440  77                       MOV   M,A               ;SAVE
8441  23                       INX   H                 ;BUMP PNTR.
```

```
8442  E3                        XTHL                    ;GET SOURCE ADD.
8443  0C                        INR     C               ;BUMP CHAR. CNT
8444  23                PK2:    INX     H               ;BUMP PNTR.
8445  05                        DCR     B               ;DCR INPUT LINE CHAR CNT
8446  C2 27 84                  JNZ     PK1             ;MORE - GO AGAIN
8449  79                        MOV     A,C             ;CHAR CNT TO A
844A  E1                        POP     H               ;RESTORE REG'S, RETURN
844B  E1                        POP     H
844C  D1                        POP     D
844D  C9                        RET
                  ; ROUTINES TO  PAD MESSAGES TO  OUTPUT BUFFER.
                  ; FOR12 PADS 'UNDERFLOW'
                  ; FOR11 PADS 'OVERFLOW'
                  ; FOR10 PADS 'ZERODIVIDE'
                  ; FORM9 PADS 'INPUT ERROR, TRY AGAIN'
                  ; FORM8 PADS 'MEMORY FULL'
                  ; FORM7 PADS 'WHAT?'
                  ; FORM4 PADS 'IN LINE'
                  ; FORM3 PADS 'ERROR'
                  ; FORM2 PADS 'TURN ON PUNCH'
                  ; FORM1 PADS 'READY'
                  ; FORM5 PADS SOURCE LINE, PASSED ADDRESS OF
                  ; LENGTH OF LINE IN HL REGS.
                  ; FORM6 PADS CHAR STRING, PASSED ADD OF FIRST CHAR IN
                  ; HL, LENGTH OF STRING IN REG C
844E  2C          FOR12:    INR     L
844F  2C          FOR11:    INR     L
8450  2C          FOR10:    INR     L
8451  2C          FORM9:    INR     L
8452  2C          FORM8:    INR     L
8453  2C          FORM7:    INR     L
8454  2C          FORM4:    INR     L
8455  2C          FORM3:    INR     L
8456  2C          FORM2:    INR     L
8457  6E          FORM1:    MOV     L,M
8458  4E          FORM5:    MOV     C,M
8459  79                    MOV     A,C
845A  FE 00                 CPI     0
845C  C8                    RZ
845D  23          F1:       INX     H
845E  7E          FORM6:    MOV     A,M
```

```
845F    06 01                       MVI     B,1
8461    CD B0 82                    CALL    PAD
8464    0D                          DCR     C
8465    C2 5D 84                    JNZ     F1
8468    C9                          RET
                        ;*****************************************
                        ;THE CODE FROM HERE TO THE NEXT LINE OF *'S MUST BE ON ONE PAGE
8469    73              @DATA:      DB      @DAT1 AND 377Q
846A    79                          DB      @DAT2 AND 377Q
846B    87                          DB      @DAT3 AND 377Q
846C    90                          DB      @DAT4 AND 377Q
846D    9A                          DB      @DAT5 AND 377Q
846E    A0                          DB      @DAT6 AND 377Q
846F    AF                          DB      @DAT7 AND 377Q
8470    C6                          DB      @DAT8 AND 377Q
8471    D1                          DB      @DAT9 AND 377Q
8472    DA                          DB      @DA10 AND 377Q
8473    05 52 45 41     @DAT1:      DB      5,'READY'
8477    44 59
8479    0D 54 55 52     @DAT2:      DB      13,'TURN ON PUNCH'
847D    4E 20 4F 4E
8481    20 50 55 4E
8485    43 48
8487    08 8D 8A 45     @DAT3:      DB      8,215Q,212Q,'ERROR '
848B    52 52 4F 52
848F    20
8490    09 20 49 4E     @DAT4:      DB      9,' IN LINE '
8494    20 4C 49 4E
8498    45 20
849A    05 57 48 41     @DAT5:      DB      5,'WHAT?'
849E    54 3F
84A0    0E 4D 45 4D     @DAT6:      DB      14,'MEMORY FULL',215Q,212Q,'?'
84A4    4F 52 59 20
84A8    46 55 4C 4C
84AC    8D 8A 3F
84AF    16 49 4E 50     @DAT7:      DB      22,'INPUT ERROR, TRY AGAIN'
84B3    55 54 20 45
84B7    52 52 4F 52
84BB    2C 20 54 52
84BF    59 20 41 47
84C3    41 49 4E
```

```
84C6  0A 49 4E 44    @DAT8:   DB    10,'INDEFINITE'
84CA  45 46 49 4E
84CE  49 54 45
84D1  08 4F 56 45    @DAT9:   DB    8,'OVERFLOW'
84D5  52 46 4C 4F
84D9  57
84DA  09 55 4E 44    @DA10:   DB    9,'UNDERFLOW'
84DE  45 52 46 4C
84E2  4F 57

                     ;*************************************************
                     ; ROUTINE TO INPUT SOURCE LINE FROM HSR. PASSED ADD
                     ; OF FIRST CHAR IN HL. RETURNS LENGTH OF LINE IN REG A
84E4  E5             HSRIN:   PUSH  H
84E5  06 00                   MVI   B,0
84E7  C3 ED 84                JMP   PIN1A
84EA  CD 17 85       PIN1:    CALL  CHAR5
84ED  FE 99          PIN1A:   CPI   231Q                ;CNTRL Y?
84EF  CA 11 84                JZ    TIN5A
84F2  FE FF                   CPI   377Q
84F4  CA 0F 85                JZ    PIN3
84F7  FE DF                   CPI   337Q
84F9  CA 0F 85                JZ    PIN3
84FC  FE 8A                   CPI   212Q
84FE  CA 1A 84                JZ    TIN4A
8501  FE 8D                   CPI   215Q
8503  CA EA 84                JZ    PIN1
8506  77                      MOV   M,A
8507  23                      INX   H
8508  04                      INR   B
8509  CD 1D 85                CALL  MEMFUL
850C  C3 EA 84                JMP   PIN1
850F  2B             PIN3:    DCX   H
8510  05                      DCR   B
8511  F2 EA 84                JP    PIN1
8514  E1                      POP   H
8515  AF                      XRA   A                  ;ZERO A
8516  C9                      RET
                     ; ROUTINE TO INPUT CHAR FROM HSR
8517  C5             CHAR5:   PUSH  B
8518  CD 0D 01                CALL  HSRDR
851B  C1                      POP   B
```

```
851C  C9                    RET
                    ; ROUTINE TO INSURE SOURCE DOES NOT OVERFLOW MEM SPACE
                    ; COMPARES CURENT MEM ADDRESS TO SP.
851D  C5            MEMFUL: PUSH  B
851E  D5                    PUSH  D
851F  E5                    PUSH  H
8520  3E 32                MVI   A,50
8522  85                    ADD   L
8523  47                    MOV   B,A
8524  3E 00                MVI   A,0
8526  8C                    ADC   H
8527  4F                    MOV   C,A
8528  21 00 00              LXI   H,0
852B  39                    DAD   SP
852C  55                    MOV   D,L
852D  5C                    MOV   E,H
852E  CD F1 97              CALL  DCOMP
8531  E1                    POP   H
8532  D1                    POP   D
8533  C1                    POP   B
8534  D0                    RNC
8535  21 69 84              LXI   H,ODATA
8538  CD 52 84              CALL  FORM8
853B  CD D3 82              CALL  WRIT
853E  CD F7 97              CALL  CHAR2
8541  CD B0 82              CALL  PAD
8544  CD D3 82              CALL  WRIT
8547  DE B0                 SBI   260Q
8549  FE 04                 CPI   4
854B  CC 9F 81              CZ    WHAT
854E  2A 00 01              LHLD  SPNT
8551  F9                    SPHL
8552  0E 01                 MVI   C,1
8554  C3 48 80              JMP   M4A
                    ; ROUTINE TO EVALUATE BOUNDS FOR LIST AND PLIST
                    ; COMMANDS. RETURNS PLINE AS FIRST LINE, KLINE
                    ; AS LAST LINE TO BE LISTED.
8557  2A 4B 04    BOUND:  LHLD  NLINE
855A  3E 09                 MVI   A,9
855C  85                    ADD   L
855D  6F                    MOV   L,A
```

```
855E  3E 00           MVI   A,0
8560  8C              ADC   H
8561  67              MOV   H,A
8562  E5              PUSH  H
8563  CD 17 82        CALL  NUMB
8566  D4 9F 81        CNC   WHAT
8569  CD 2D 82        CALL  CVB
856C  F5              PUSH  PSW
856D  C5              PUSH  B
856E  CD AD 85        CALL  BND2
8571  C1              POP   B
8572  2B              DCX   H
8573  22 59 04        SHLD  PLINE
8576  F1        BND1: POP   PSW
8577  E1              POP   H
8578  3C              INR   A
8579  85              ADD   L
857A  6F              MOV   L,A
857B  3E 00           MVI   A,0
857D  8C              ADC   H
857E  67              MOV   H,A
857F  3E 00           MVI   A,0
8581  B9              CMP   C
8582  C8              RZ
8583  0D              DCR   C
8584  CD 17 82        CALL  NUMB
8587  D4 9F 81        CNC   WHAT
858A  D5              PUSH  D
858B  CD 2D 82        CALL  CVB
858E  D5              PUSH  D
858F  C5              PUSH  B
8590  CD AD 85        CALL  BND2
8593  C1              POP   B
8594  23              INX   H
8595  56              MOV   D,M
8596  23              INX   H
8597  5E              MOV   E,M
8598  EB              XCHG
8599  22 52 04        SHLD  KLINE
859C  D1              POP   D
859D  E1              POP   H
```

```
859E  79                      MOV   A,C
859F  FE 00                   CPI   0
85A1  C2 9F 81                JNZ   WHAT
85A4  44                      MOV   B,H
85A5  4D                      MOV   C,L
85A6  CD F1 97                CALL  DCOMP
85A9  D0                      RNC
85AA  C3 9F 81                JMP   WHAT
85AD  2A 49 04          BND2: LHLD  STLINE
85B0  46                BND3: MOV   B,M
85B1  23                      INX   H
85B2  4E                      MOV   C,M
85B3  CD F1 97                CALL  DCOMP
85B6  D8                      RC
85B7  C8                      RZ
85B8  E5                      PUSH  H
85B9  23                      INX   H
85BA  7E                      MOV   A,M
85BB  23                      INX   H
85BC  66                      MOV   H,M
85BD  6F                      MOV   L,A
85BE  CD A6 82                CALL  CHK1
85C1  C1                      POP   B
85C2  D2 B0 85                JNC   BND3
85C5  C5                      PUSH  B
85C6  E1                      POP   H
85C7  C9                      RET
                        ; ROUTINE TO OUTPUT ERROR MSG. TO USER.
                        ; REG A CONTAINS BCD ERROR NUMBER, HL
                        ; LOADED WITH VALUE OF KLINE.
85C8  21 13 80          ERROR: LXI    H,M1A           ;RETURN ADDRESS
85CB  E5                      PUSH  H               ;PUT ON STACK
85CC  21 69 84                LXI   H,ODATA         ;OUTPUT BUFFER DATA TABLES
85CF  E5                      PUSH  H
85D0  57                      MOV   D,A             ;SAVE ERROR NUMB. IN D
85D1  CD 55 84                CALL  FORM3           ;PAD 'ERROR '
85D4  06 01                   MVI   B,1             ;INIT FOR PADS
85D6  48                      MOV   C,B             ;INIT AS CNTR.
85D7  7A                      MOV   A,D             ;GET ERROR NUMB.
85D8  07                      RLC                   ;ROTATE HIGH 4 BITS TO LOW 4
85D9  07                      RLC
```

```
85DA  07                  RLC
85DB  07                  RLC
85DC  E6 0F       ERRR1:  ANI   17Q             ;MASK
85DE  C6 B0               ADI   260Q            ;CONVERT TO ASCII
85E0  CD B0 82            CALL  PAD             ;PAD IT
85E3  7A                  MOV   A,D             ;GET ERROR NUMB.
85E4  0D                  DCR   C               ;ANOTHER PASS?
85E5  F2 DC 85            JP    ERRR1           ;YES
85E8  E1                  POP   H               ;NO-CONTINUE
85E9  CD 54 84    ERLN:   CALL  FORM4
85EC  2A 52 04            LHLD  KLINE
85EF  23                  INX   H
85F0  23                  INX   H
85F1  23                  INX   H
85F2  23                  INX   H
85F3  4E                  MOV   C,M
85F4  23                  INX   H
85F5  CD 74 82            CALL  LENGTH
85F8  4F                  MOV   C,A
85F9  CD 5E 84            CALL  FORM6
85FC  CD D3 82            CALL  WRIT
85FF  C9                  RET
                  ;THIS ROUTINE INCREMENTS H AND L AND
                  ;DECR. C(CHARS IN LINE) SHOULD C RESULT
                  ;IN 0 THEN THE ERROR CORRES. TO ENTRY PNT.
                  ;IS GIVEN
8600  3E 07       ICP7:   MVI   A,7
8602  C3 11 86            JMP   INCPT
8605  3E 08       ICP8:   MVI   A,8
8607  C3 11 86            JMP   INCPT
860A  3E 04       ICP4:   MVI   A,4
860C  C3 11 86            JMP   INCPT
860F  3E 02       ICP2:   MVI   A,2
8611  23          INCPT:  INX   H
8612  0D                  DCR   C
8613  C0                  RNZ
8614  C3 C8 85            JMP   ERROR
                  ;FSYM FINDS SYMBOLS IN TABLE
                  ;B,C CONTAIN SYMBOL
                  ;RET WITH B,C,D,E SAME
                  ;H AND L PNT TO VALUE (1ST BYTE)
```

```
                        ;CY=1   =@ FOUND
                        ;CY=0   AND A SCALAR VAR. =@ INSERTED
                        ;       AND SET TO 0
                        ;CY=0   AND AN ARRAY   =@ NO ACTION,
                        ;       H AND L PNT TO LAST ENTRY IN SYMBOL TABLE
8617    D5              FSYM:   PUSH    D
8618    AF                      XRA     A
8619    B0                      ORA     B               ;SET CARRY IF NOT
861A    CA 1E 86                JZ      AR              ;AN ARRAY AND SAVE
861D    3F                      CMC
861E    F5              AR:     PUSH    PSW
861F    2A 59 04                LHLD    NXTSP           ;GET NEXT AVAILABLE
8622    C5                      PUSH    B               ;SPACE PNTR.
8623    44                      MOV     B,H
8624    4D                      MOV     C,L             ;CHECK TO SEE
8625    2A 4B 04                LHLD    STSPAC          ;IF SYMBOL TABLE
8628    54                      MOV     D,H             ;EMPTY
8629    5D                      MOV     E,L
862A    CD F1 97                CALL    DCOMP           ;DOUBLE BYTE COMPARE
862D    C1                      POP     B               ;GET VAR. BACK
862E    CA 4E 86                JZ      NOSYM
8631    CD A6 82        LUKON:  CALL    CHK1            ;CHECK FOR END
8634    DA 50 86                JC      NOENT
8637    54                      MOV     D,H             ;SAVE OLD PNTR
8638    5D                      MOV     E,L
8639    78                      MOV     A,B
863A    BE                      CMP     M               ;DO VARIABLES MATCH
863B    C2 45 86                JNZ     NOMAT
863E    23                      INX     H
863F    79                      MOV     A,C
8640    BE                      CMP     M
8641    CA 7E 86                JZ      ENTRY
8644    2B                      DCX     H
8645    23              NOMAT:  INX     H               ;NO MATCH GET NEW PNT.
8646    23                      INX     H
8647    7E                      MOV     A,M
8648    23                      INX     H
8649    66                      MOV     H,M
864A    6F                      MOV     L,A
864B    C3 31 86                JMP     LUKON
                        ;ARRIVE HERE IF SYMBOL TABLE IS EMPTY
```

```
864E  1B           NOSYM:   DCX   D          ; =STSPAC-2 SO STPNT WORKS RIGHT
864F  1B                    DCX   D
                   ;ARRIVE HERE WHEN NO ENTRY FOUND
8650  2A 59 04     NOENT:   LHLD  NXTSP      ;ADD. OF FREE MEMORY
8653  EB                    XCHG              ;TO DE, HL HAVE LAST SYM. TAB.
8654  F1                    POP   PSW         ;ENTRY ARRAY?
8655  D2 83 86              JNC   FBAC        ;YES, RETURN
8658  CD BA 8F              CALL  CHKLC       ;CHECK FOR PAGE BOUNDARY CROSSING
865B  CD 8C 81              CALL  STPNT       ;UPDATE PNTR
865E  EB                    XCHG              ;NXTSP TO HL
865F  70                    MOV   M,B         ;STORE VAR.
8660  23                    INX   H
8661  71                    MOV   M,C
8662  23                    INX   H
8663  E5                    PUSH  H
8664  23                    INX   H           ;STORE NXTSP+8 IN NXTSP
8665  23                    INX   H
8666  23                    INX   H
8667  23                    INX   H
8668  23                    INX   H
8669  23                    INX   H
866A  22 59 04              SHLD  NXTSP
866D  CD 1D 85              CALL  MEMFUL      ;MEMORY FULL?
8670  E1                    POP   H           ;SET FWD PNT. TO -1
8671  36 FF                 MVI   M,377Q
8673  23                    INX   H
8674  36 FF                 MVI   M,377Q
8676  23                    INX   H           ;INIT TO FLT. PNT. 0
8677  CD CD 97              CALL  ZROL
867A  B7                    ORA   A           ;CLEAR CY
867B  C3 83 86              JMP   FBAC        ;RESET CARRY AND RETURN
867E  F1           ENTRY:   POP   PSW         ;VAR FOUND
867F  23                    INX   H           ;MOVE PNT. TO FIRST BYTE
8680  23                    INX   H           ;OF FLT. PNT. NO.
8681  23                    INX   H
8682  37                    STC               ;SET CY AND RET.
8683  D1           FBAC:    POP   D           ;RESTORE D
8684  C9                    RET
                   ;
                   ;
                   ;RUN - THE INTERP.
```

```
                        ;
                        ;INIT. NXTSP
8685    2A 4B 04    RUN:    LHLD    STSPAC
8688    EB              XCHG
8689    CD CB 8F        CALL    CKDIM       ; ADJUST START OF SYMBOL TABLE SO
                                            ; IT STARTS ON AN EVEN 4 WORD BOUNDARY
868C    CD BA 8F        CALL    CHKLC       ; ADJUST START OF SYMBOL TABLE SO IT
                                            ; DOES NOT CROSS PAGE BOUNDARY
868F    EB              XCHG
8690    22 4B 04        SHLD     STSPAC
8693    22 59 04        SHLD    NXTSP
8696    21 AA 04        LXI     H,BOTNS     ;INIT SP FOR NESTING STACK
8699    22 94 04        SHLD    NEST
869C    21 13 80        LXI     H,M1A       ;PRECAUTION, IN CASE RETURN IS
869F    E5              PUSH    H           ;EXECUTED BEFORE A GOSUB
86A0    E5              PUSH    H
86A1    2A 49 04        LHLD    STLINE      ;START OF SOURCE
86A4    CD 09 82    ILOOP:  CALL    QUITT       ;CHECK FOR INTERRUPTION
86A7    CD A6 82        CALL    CHK1        ;HL=-1 =@ NO MORE SOURCE
86AA    D2 B2 86        JNC     SORCE
86AD    3E 01           MVI     A,1
86AF    C3 C8 85        JMP     ERROR       ;ERROR 1, NO END STMT.
86B2    22 52 04    SORCE:  SHLD    LPNT
86B5    E5              PUSH    H
86B6    21 52 04        LXI     H,LPNT      ;DEFINE VALUES OF
86B9    CD EE 97        CALL    PTVAL       ;KBIN,KFPNT,KLEN
86BC    3A 58 04        LDA     KLEN        ;CHAR'S IN LINE TO C
86BF    4F              MOV     C,A
86C0    0C              INR     C
86C1    E1              POP     H           ;MOVE PNTR. TO 1ST CHAR
86C2    23              INX     H           ;IN SOURCE REC.
86C3    23              INX     H
86C4    23              INX     H
86C5    23              INX     H
86C6    CD 0F 86    L1:     CALL    ICP2        ;INCR. H,L DCR C
86C9    CD 25 82        CALL    ALPHA       ;FIND FIRST LETTER
86CC    D2 C6 86        JNC     L1
86CF    AF              XRA     A
86D0    3C              INR     A           ;LETTER FOUND
86D1    CD 00 83        CALL    SYMSRT      ;DETERMINE KEYWORD
```

```
86D4  FE FF           CPI   377Q
86D6  C2 DE 86        JNZ   GKEY
86D9  3E 02           MVI   A,2          ;BAD KEYWORD
86DB  C3 C8 85        JMP   ERROR
86DE  22 5B 04  GKEY: SHLD  CPNT
86E1  21 EE 86        LXI   H,JTBL       ;LOAD JUMP TABLE PNTR.
86E4  87              ADD   A            ;DOUBLE A
86E5  5F              MOV   E,A
86E6  16 00           MVI   D,0
86E8  19              DAD   D            ;PNT. TO PROPER PROC.
86E9  7E              MOV   A,M          ;ADD. IN JUMP TABLE
86EA  23              INX   H            ;GET PROC. ADD.
86EB  66              MOV   H,M
86EC  6F              MOV   L,A
86ED  E9              PCHL               ;INDIRECT JUMP TO PROC.
86EE  FD 8B     JTBL: DW    LET          ;JMP TABLE
86F0  B5 8A           DW    PRI
86F2  AC 8B           DW    IEND         ;REM STMT. - NO ACTION
86F4  13 80           DW    M1A          ;STOP STMT.-RETURN TO EDIT MODE
86F6  0A 87           DW    ENDD
86F8  18 87           DW    GOTO
86FA  9D 8C           DW    IFRT
86FC  6C 8B           DW    INPUT
86FE  38 87           DW    DIM
8700  7A 8D           DW    CALLP
8702  F0 8D           DW    GOSUB
8704  05 8E           DW    RETRN
8706  07 8E           DW    FOR
8708  F9 8E           DW    NEXT
870A  2A 56 04  ENDD: LHLD  KFPNT        ;CHECK TO SEE IF MORE
870D  CD A6 82        CALL  CHK1         ;SOURCE AFTER END
8710  DA 13 80        JC    M1A
8713  3E 03           MVI   A,3          ;MORE SOURCE ERROR 3
8715  C3 C8 85        JMP   ERROR
8718  2A 5B 04  GOTO: LHLD  CPNT         ;GOTO STMT. PROC.
871B  23     GSENT: INX   H              ;INCREMENT PAST KEYWORD
871C  23            INX   H
871D  23            INX   H
871E  CD 0A 86        CALL  ICP4         ;POSSIBLE ERROR 4
8721  CD 2D 82  GTRA: CALL  CVB          ;GET DESTINATION
8724  B7              ORA   A            ;MAKE SURE IT WAS OK
```

```
8725  C2 2D 87            JNZ   ØKN
8728  3E 04               MVI   A,4
872A  C3 C8 85            JMP   ERRØR
872D  CD 8B 82    ØKN:    CALL  NSRCH       ;GET NEXT LPNT
8730  D2 A4 86            JNC   ILØØP       ;MAKE SURE IT EXISTED
8733  3E 05               MVI   A,5
8735  C3 C8 85            JMP   ERRØR       ;NØN-EXISTENT
8738  2A 5B 04    DIM:    LHLD  CPNT        ;DIM STMT. PRØC.
873B  23                  INX   H           ;PNT TØ FIRST VAR.
873C  23                  INX   H
873D  23                  INX   H
873E  CD 25 82    DLØØP:  CALL  ALPHA       ;CHECK IF IT IS A VAR.
8741  DA 49 87            JC    ØKLET
8744  3E 06       ER6:    MVI   A,6         ;ERRØR 6
8746  C3 C8 85            JMP   ERRØR
8749  46          ØKLET:  MØV   B,M
874A  CD 00 86            CALL  ICP7        ;INCR.CPNT
874D  3E A8               MVI   A,250Q      ;CHECK FØR (
874F  BE                  CMP   M
8750  C2 44 87            JNZ   ER6
8753  CD 00 86            CALL  ICP7        ;INCR. CPNT
8756  CD 2D 82            CALL  CVB         ;CØNV. TØ BIN NØ.
8759  85                  ADD   L           ;UPDATE CPNT
875A  6F                  MØV   L,A         ;ED CØNTAIN ARRAY LEN.
875B  3E 00               MVI   A,0
875D  8C                  ADC   H           ;C CØNT. NØ. CHARS LEFT
875E  67                  MØV   H,A         ;IN LINE
875F  3E A9               MVI   A,251Q      ;CHECK FØR )
8761  BE                  CMP   M
8762  C2 44 87            JNZ   ER6
8765  E5                  PUSH  H
8766  C5                  PUSH  B           ;SAVE B,C,H,L
8767  48                  MØV   C,B         ;SET UP FØR CALL TØ FSYM
8768  06 00               MVI   B,0
876A  CD 17 86            CALL  FSYM
876D  D2 77 87            JNC   NDØU
8770  C1                  PØP   B
8771  E1                  PØP   H
8772  3E 11               MVI   A,11H       ;ERRØR 11
8774  C3 C8 85            JMP   ERRØR       ;DUPLICATE ARRAY DEF.
8777  D5          NDØU:   PUSH  D           ;SAVE DIM. LENGTH
```

```
8778  EB                    XCHG
8779  2A 59 04              LHLD    NXTSP       ;ADD. OF LAST SYM. TAB. ENTRY TO DE
877C  EB                    XCHG                ;GET ADD. OF AVAILABLE MEM.
877D  CD CB 8F              CALL    CKDIM       ;SET UP FOR CALL
8780  CD 8C 81              CALL    STPNT       ;  CHECK START OF 'DIM' ARRAY
8783  EB                    XCHG                ;STORE NEW PNTR
8784  D1                    POP     D           ;NXTSP TO HL
8785  36 00                 MVI     M,0         ;RESTORE D
8787  23                    INX     H
8788  71                    MOV     M,C         ;INSERT VAR IN SYMB. TAB.
8789  23                    INX     H
878A  36 FF                 MVI     M,377Q      ;FPNT TO -1
878C  23                    INX     H
878D  36 FF                 MVI     M,377Q
878F  23                    INX     H           ;PNTS TO FIRST DATA
8790  7A                    MOV     A,D         ;GET ONE'S COMPLEMENT OF
8791  2F                    CMA                 ;NUMBER OF ELEMENTS
8792  4F                    MOV     C,A         ;IN ARRAY TO B,C
8793  7B                    MOV     A,E
8794  2F                    CMA
8795  47                    MOV     B,A
8796  CD CD 97        CONT: CALL    ZROL        ;ZEROE OUT ELEMTS.
8799  23                    INX     H           ; OF ARRAY
879A  23                    INX     H
879B  23                    INX     H
879C  23                    INX     H
879D  03                    INX     B
879E  E5                    PUSH    H
879F  CD 1D 85              CALL    MEMFUL      ;MEMORY FULL?
87A2  60                    MOV     H,B
87A3  69                    MOV     L,C
87A4  CD A6 82              CALL    CHK1
87A7  E1                    POP     H
87A8  D2 96 87              JNC     CONT
87AB  22 59 04              SHLD    NXTSP       ;NEW VALUE OF NXTSP.
87AE  C1                    POP     B           ;RESTORE REG'S
87AF  E1                    POP     H
87B0  23                    INX     H
87B1  0D                    DCR     C
87B2  CA AC 8B              JZ      IEND        ;MORE ELEMTS IN LINE?
87B5  0D                    DCR     C
```

```
87B6  CA 44 87              JZ      ER6
87B9  3E AC                 MVI     A,254Q          ;NEXT ELEMENT A ,
87BB  BE                    CMP     M
87BC  23                    INX     H
87BD  CA 3E 87              JZ      DLOOP
87C0  C3 44 87              JMP     ER6
                    ;ROUTINE TO COPY CONTENTS PNTED TO
                    ;BY DE TO LOCATION H,L
87C3  F5            COPDH:  PUSH    PSW             ;SAVE REGISTERS
87C4  C5                    PUSH    B
87C5  D5                    PUSH    D
87C6  E5                    PUSH    H
87C7  06 04                 MVI     B,4             ;COUNT
87C9  1A            COPD1:  LDAX    D               ;GET FROM SOURCE
87CA  77                    MOV     M,A             ;PUT TO DESTINATION
87CB  13                    INX     D               ;BUMP PNTRS, CNT
87CC  23                    INX     H
87CD  05                    DCR     B
87CE  C2 C9 87              JNZ     COPD1
87D1  E1                    POP     H               ;RESTORE REGISTERS
87D2  D1                    POP     D
87D3  C1                    POP     B
87D4  F1                    POP     PSW
87D5  C9                    RET
                    ;OUTR PADS OUTPUT FROM CONV INTO
                    ;OUTPUT BUFFER USING ROUTINE PAD
                    ;ALL REG'S MAINTAINED
87D6  C5            OUTR:   PUSH    B               ;SAVE REG B
87D7  06 01                 MVI     B,1             ;PAD ONCE
87D9  CD B0 82              CALL    PAD             ;DO IT
87DC  C1                    POP     B               ;RESTORE B AND RET.
87DD  C9                    RET
                    ;VALUE RETURNS IN D(H),E(L) PNTR.
                    ;TO THE VALUE OF A TOKEN
                    ;C,H,L ARE UPDATED
                    ;A,B ARE DESTROYED
87DE  CD 8D 88      VALUE:  CALL    VAR             ;IS IT A VARIABLE?
87E1  D8                    RC                      ;YES - ALL DONE
87E2  3E 03                 MVI     A,3             ;NO CHEK IF A FUNC.
87E4  CD 00 83              CALL    SYMSRT
87E7  FE FF                 CPI     377Q
```

```
87E9  CA 56 88          JZ      KONT            ;NOT A FUNCTION -
87EC  FE 01             CPI     1               ;WAS IT PUT(--)?
87EE  C2 F4 87          JNZ     GET             ;NO - OK
87F1  C3 98 8C          JMP     ER10            ;ILLEGAL USE OF FUNCTION
87F4  23        GET:    INX     H               ;OK, IT'S GET(--)
87F5  23                INX     H               ;UPDATE H,L
87F6  23                INX     H
87F7  79                MOV     A,C             ;CHECK FOR PREMATURE EOL
87F8  B7                ORA     A
87F9  CA 42 8A          JZ      ER8
87FC  3E A8             MVI     A,250Q          ;CHEK FOR (
87FE  BE                CMP     M
87FF  C2 42 8A          JNZ     ER8
8802  CD 05 86          CALL    ICP8            ;BUMP PNTR'S
8805  CD E7 89          CALL    EVAL            ;GET PORT #
8808  E5                PUSH    H               ;SAVE REG H,L
8809  21 7C 04          LXI     H,FREG1
880C  CD C3 87          CALL    COPDH           ;COPY IT
880F  EB                XCHG
8810  E1                POP     H               ;RESTORE H,L
8811  CD 35 89          CALL    FIX             ;FIX IT
8814  13                INX     D
8815  13                INX     D               ;GET LOWEST BYTE TO
8816  13                INX     D               ;REG D
8817  1A                LDAX    D
8818  57                MOV     D,A
8819  79                MOV     A,C             ;EOL?
881A  B7                ORA     A
881B  CA 42 8A          JZ      ER8
881E  3E A9             MVI     A,251Q          ;CHECK FOR )
8820  BE                CMP     M
8821  C2 42 8A          JNZ     ER8
8824  23                INX     H               ;BUMP PNTR'S
8825  0D                DCR     C
8826  E5                PUSH    H               ;SAVE H,L,B,C
8827  C5                PUSH    B               ;STORE PROGRAM SEGMENT
8828  01 77 04          LXI     B,GREG          ;IN RAM,START AT GREG
882B  21 51 88          LXI     H,RINST         ;ADD. OF INST'S
882E  1E 05             MVI     E,5             ;NUMB. OF BYTES
8830  7E        V1:     MOV     A,M             ;GET BYTE
8831  02                STAX    B               ;STORE IN RAM
```

```
8832  23                    INX   H
8833  03                    INX   B
8834  1D                    DCR   E                 ;BUMP PNTR'S,DCR CNT
8835  C2 30 88              JNZ   V1
8838  21 78 04              LXI   H,GREG+1          ;STORE PORT =
883B  72                    MOV   M,D               ;IN RAM
883C  C3 77 04              JMP   GREG              ;OK - TRANSFER
883F  21 79 04        HOME: LXI   H,GREG+2          ;SET UP FOR FLOAT
8842  77                    MOV   M,A               ;STORE AWAY INPUT
8843  2B                    DCX   H
8844  AF                    XRA   A                 ;ZERO OUT HIGHER BYTES
8845  77                    MOV   M,A               ;BUT CHAR. DOESN'T MATTER
8846  2B                    DCX   H
8847  77                    MOV   M,A
8848  CD DC 97              CALL  DFXL              ;FLOAT IT
884B  11 77 04              LXI   D,GREG            ;FIX D,E RESTORE C,H,L
884E  C1                    POP   B
884F  E1                    POP   H
8850  C9                    RET
8851  DB 00           RINST: IN   0                 ;RAM INSTRUCTIONS
8853  C3 3F 88              JMP   HOME
8856  CD 17 82        KONT: CALL  NUMB              ;NUMBER
8859  DA 62 88              JC    OKK
885C  3E AE                 MVI   A,256Q            ;DEC. PNT.?
885E  BE                    CMP   M
885F  C2 42 8A              JNZ   ER8
8862  3E 01           OKK:   MVI   A,1             ;MODE=1, IE. INPUT FROM SOURCE
8864  CD 6E 88              CALL  RDKON             ;READ CONSTANT TO GREG
8867  DA 88 88              JC    ER9               ;IF ERROR THEN CY=1
886A  11 77 04              LXI   D,GREG            ;PNTS. TO CONSTANT
886D  C9                    RET
                  ;THIS ROUTINE READS A CONSTANT INTO GREG FROM ASCII
                  ;CHARACTERS POINTED TO BY HL AND C
                  ;ENTER WITH A=0 => DATA FROM TTY
                  ;ENTER WITH A=1 => DATA FROM SOURCE
                  ;RETURN WITH CY=1 => ERROR IN CONVERSION
886E  32 85 04  RDKON: STA   MODE              ;SAVE MODE FOR ROUT. INP
8871  22 86 04        SHLD  HLINP               ;SAVE HL FOR ROUT. INP
8874  79              MOV   A,C
8875  32 84 04        STA   CREG                ;SAVE C FOR ROUT. INP
8878  21 77 04        LXI   H,GREG              ;WHER VALUE WILL GO
```

```
887B  0E 66              MVI   C,SCR AND 377Q   ;SET UP AND CALL FINPT
887D  CD E8 97           CALL  FINPT
8880  2A 86 04           LHLD  HLINP            ;RETORE H,L AND C
8883  3A 84 04           LDA   CREG
8886  4F                 MOV   C,A
8887  C9                 RET                    ;DONE
8888  3E 09        ER9:  MVI   A,9
888A  C3 C8 85           JMP   ERROR
                 ;VAR DECIDES WHETHER A TOKEN IS
                 ;A VARIABLE IF SO CY=1 AND
                 ;ADDRESS IS COMPUTED,(SUBSCRIPT IS
                 ;EVALUATED ETC.), RETURNS WITH DE PNTING
                 ;TO VAR. REFERENCED H,L,C,UPDATED
                 ;A,B DESTROYED
                 ;IF NOT A VARIBLE CY=0
                 ;H,L,C ARE LEFT UNTOUCHED
888D  CD 25 82   VAR:  CALL  ALPHA              ;1ST CHAR A LETTER?
8890  D0               RNC                      ;NO-NOT VAR.
8891  23               INX   H                  ;BUMP PNTR'S
8892  0D               DCR   C
8893  C2 9F 88         JNZ   MORE               ;MORE TO LINE
8896  C5         SC1:  PUSH  B                  ;SAVE B,EOL
8897  0E 00            MVI   C,0                ;SET FOR CALL TO FSYM
8899  2B               DCX   H                  ;GET SINGLE LETTER
889A  46               MOV   B,M                ;VAR TO B
889B  23               INX   H
889C  C3 D1 88         JMP   SCALR
889F  CD 25 82   MORE: CALL  ALPHA              ;2ND A LETTER?
88A2  D2 B4 88         JNC   SFSG               ;SO FAR SO GOOD
88A5  C5               PUSH  B                  ;SAVE C
88A6  3E 02            MVI   A,2                ;CHECK FOR DELIMITER
88A8  CD 00 83         CALL  SYMSRT
88AB  C1               POP   B                  ;RESTORE C
88AC  3C               INR   A                  ;FOUND?
88AD  C2 96 88         JNZ   SC1                ;YES
88B0  0C         BUPT: INR   C                  ;NOT A VAR.
88B1  2B               DCX   H                  ;BACK UP PNTR'S
88B2  B7               ORA   A                  ;CY=0 AND RET
88B3  C9               RET
88B4  CD 17 82   SFSG: CALL  NUMB               ;TEST FOR NUMBER
88B7  D2 D9 88         JNC   ARCK               ;MAYBE AN ARRAY
```

```
88BA  23                  INX   H          ;ITS A SCALAR
88BB  0D                  DCR   C          ;BUMP PNTR'S
88BC  CA CA 88            JZ    SLOAD      ;EOL
88BF  C5                  PUSH  B          ;SAVE C
88C0  3E 02               MVI   A,2        ;SET UP FOR SYMSRT
88C2  CD 00 83            CALL  SYMSRT     ;TEST FOR LEGAL
88C5  C1                  POP   B          ;GET C BACK
88C6  3C                  INR   A          ;DELIMITER FOUND?
88C7  CA 42 8A            JZ    ER8        ;NO, ERROR
88CA  2B            SLOAD: DCX   H          ;MOVE BACK,
88CB  C5                  PUSH  B          ;SAVE C,
88CC  4E                  MOV   C,M        ;GET VAR. INTO
88CD  2B                  DCX   H          ;B,C FOR FSYM
88CE  46                  MOV   B,M
88CF  23                  INX   H
88D0  23                  INX   H
88D1  EB            SCALR: XCHG
88D2  CD 17 86            CALL  FSYM       ;SAVE H,L IN D,E
88D5  EB                  XCHG             ;GET PNTR TO VALUE
88D6  C1                  POP   B          ;RESTORE H,L PNTR TO DE
88D7  37                  STC              ;GET C REG BACK
88D8  C9                  RET              ;SET CY,RET
88D9  7E            ARCK: MOV   A,M        ;ARRAY CHEK, GET CHARACTER
88DA  FE A8               CPI   250Q       ;IS IT (?
88DC  CA ED 88            JZ    ARYES      ;YES,ITS AN ARRAY
88DF  3E 02               MVI   A,2        ;NO-CHEK FOR LEGAL DELIM.
88E1  C5                  PUSH  B          ;SAVE C
88E2  CD 00 83            CALL  SYMSRT
88E5  C1                  POP   B          ;RESTORE C
88E6  3C                  INR   A          ;DELIMITER FOUND?
88E7  CA 42 8A            JZ    ER8
88EA  C3 96 88            JMP   SC1        ;1 CHAR. SCALAR VAR.
88ED  2B            ARYES: DCX   H          ;YES-WE HAVE ARRAY
88EE  7E                  MOV   A,M        ;GET VAR.
88EF  23                  INX   H
88F0  F5                  PUSH  PSW        ;SAVE VAR.
88F1  CD 05 86            CALL  ICP8       ;BUMP PNTR'S
88F4  CD E7 89            CALL  EVAL       ;EVALUATE SUBSCRIPT
88F7  E5                  PUSH  H          ;SAVE REG H,L
88F8  21 7C 04            LXI   H,FREG1
88FB  CD C3 87            CALL  COPDH      ;COPY IT
```

```
88FE   EB                  XCHG
88FF   E1                  POP     H               ;RESTORE H,L
8900   CD 35 89            CALL    FIX             ;FIX VALUE
8903   3E A9               MVI     A,251Q          ;CHECK FOR )
8905   BE                  CMP     M
8906   C2 42 8A            JNZ     ER8
8909   23                  INX     H
890A   0D                  DCR     C               ;BUMP PNTR'S
890B   13                  INX     D               ;PNT TO LOWER 2 BYTES
890C   13                  INX     D
890D   1A                  LDAX    D
890E   47                  MOV     B,A             ;H-BYTE TO B
890F   13                  INX     D               ;PNT TO LOW BYTE
8910   1A                  LDAX    D               ;LOW BYTE TO A
8911   B7                  ORA     A               ;KILL CY
8912   17                  RAL                     ;START MULT OF OFFSET
8913   5F                  MOV     E,A             ;BY 4(BYTES/FLTPT =)
8914   78                  MOV     A,B             ;GET H BYTE
8915   17                  RAL
8916   57                  MOV     D,A             ;DE IS OFFSET*2
8917   7B                  MOV     A,E             ;GET LOW
8918   B7                  ORA     A               ;KILL CARRY
8919   17                  RAL
891A   5F                  MOV     E,A
891B   7A                  MOV     A,D
891C   17                  RAL
891D   57                  MOV     D,A
891E   F1                  POP     PSW             ;DE CONTAIN OFFSET*4
891F   C5                  PUSH    B               ;GET VAR., SAVE C
8920   4F                  MOV     C,A
8921   06 00               MVI     B,0             ;SETUP TO CALL FSYM
8923   E5                  PUSH    H               ;SAVE H,L
8924   CD 17 86            CALL    FSYM            ;GET START ADD.
8927   DA 2F 89            JC      AFOND
892A   3E 12               MVI     A,12H           ;ERROR 12
892C   C3 C8 85            JMP     ERROR           ;ARRAY REF. NOT DIM'ED.
892F   19          AFOND:  DAD     D               ;H,L NOW PNT TO START OF
8930   EB                  XCHG                    ;ARRAY, ADD OFFSET, EXCHG
8931   E1                  POP     H               ;RESTORE PNTR'S AND RET.
8932   C1                  POP     B
8933   37                  STC                     ;SET CY
```

```
8934    C9                      RET
                        ;ROUTINE TO FIX FLOATING POINT
                        ;NUMBERS, ALL REG'S BUT A ARE
                        ;MAINTAINED. DE PNT TO 4 BYTES
                        ;OF = TO BE FIXED
8935    C5              FIX:    PUSH    B
8936    E5                      PUSH    H
8937    D5                      PUSH    D               ;SAVE REG'S
8938    13                      INX     D
8939    13                      INX     D
893A    13                      INX     D               ;PNT TO 4TH BYTE
893B    1A                      LDAX    D
893C    F5                      PUSH    PSW             ;SAVE CHAR. (FOR SIGN)
893D    E6 7F                   ANI     177Q
893F    17                      RAL                     ;CHEK IF EXP SIGN IS -
8940    17                      RAL
8941    DA 50 89                JC      MINSE
8944    1F                      RAR
8945    1F                      RAR                     ;RESTORE CHAR
8946    FE 18                   CPI     30Q             ;IS IT TOO BIG?
8948    DA 52 89                JC      GOOD
894B    3E 13                   MVI     A,13H           ;ERROR 13
894D    C3 C8 85                JMP     ERROR           ;FIX = TOO BIG
8950    1F              MINSE:  RAR
8951    1F                      RAR
8952    12              GOOD:   STAX    D               ;ABSOLUTE VALUE
8953    1B                      DCX     D
8954    1B                      DCX     D
8955    1B                      DCX     D               ;MOV PNTR BACK
8956    21 7C 04                LXI     H,FREG1
8959    CD C3 87                CALL    COPDH           ;COPY TO FREG1
895C    21 80 04                LXI     H,FREG2         ;STORE .5*2**24 IN
895F    11 8F 89                LXI     D,FDAT          ;FREG2
8962    CD C3 87                CALL    COPDH           ;COPY IT
8965    21 7C 04                LXI     H,FREG1         ;SET UP TO CALL LADD
8968    06 80                   MVI     B,FREG2 AND 377Q
896A    0E 66                   MVI     C,SCR AND 377Q
896C    CD D0 97                CALL    LADD            ;ADD THEM,RESULT IN FREG1
896F    21 7C 04                LXI     H,FREG1
8972    F1                      POP     PSW             ;GET SIGN AND ADD.
8973    D1                      POP     D
```

```
8974    17                          RAL
8975    3E 00                       MVI     A,0             ;GET SIGN ONLY
8977    1F                          RAR
8978    46                          MOV     B,M             ;GET BYTE1
8979    12                          STAX    D               ;STORE BYTE 1 OF FIX
897A    78                          MOV     A,B
897B    E6 7F                       ANI     177Q            ;CLEAR HIGH BIT (FROM ADD)
897D    13                          INX     D
897E    23                          INX     H
897F    46                          MOV     B,M             ;GET BYTE 2
8980    12                          STAX    D               ;STORE BYTE 2 OF FIX
8981    13                          INX     D
8982    78                          MOV     A,B
8983    23                          INX     H
8984    46                          MOV     B,M             ;GET BYTE 3
8985    12                          STAX    D               ;STORE BYTE 3 OF FIX
8986    78                          MOV     A,B
8987    13                          INX     D
8988    12                          STAX    D               ;STORE BYTE 4 OF FIX
8989    1B                          DCX     D               ;FIX D PNTR
898A    1B                          DCX     D
898B    1B                          DCX     D
898C    E1                          POP     H
898D    C1                          POP     B
898E    C9                          RET
898F    80 00 00 18     FDAT:       DB      200Q,0,0,30Q
                        ;INP SAVES ALL REG'S
                        ;SERVES AS BUFFER BETWEEN FINPT AND
                        ;DATA INPUT. IF MODE=0, DATA COMES FROM TTY
                        ;IF MODE=1 DATA COMES FROM SOURCE STMTS.
                        ;IN ALL CASES HL,C ARE UPDATED FROM HLINP, AND
                        ;CREG AND RETURNED TO THOSE LOCATIONS
8993    E5              INP:        PUSH    H               ;SAVE ALL REG'S
8994    D5                          PUSH    D
8995    C5                          PUSH    B
8996    2A 86 04                    LHLD    HLINP           ;GET PNTR'S
8999    3A 84 04                    LDA     CREG
899C    4F                          MOV     C,A
899D    B7                          ORA     A               ;CHECK FOR EOL
899E    C2 A7 89                    JNZ     CHKMD           ;NO CHECK MODE
89A1    3E A0           SPACE:      MVI     A,240Q          ;SEND A SPACE
```

```
89A3  C1           IDONE:  POP   B              ;RESTORE REG'S
89A4  D1                   POP   D
89A5  E1                   POP   H
89A6  C9                   RET                  ;AND RETURN
89A7  3A 85 04     CHKMD:  LDA   MODE           ;GET MODE
89AA  3D                   DCR   A              ;CHECK IT
89AB  CA B7 89             JZ    MODE1          ;MODE IS 1
89AE  7E                   MOV   A,M            ;MODE 0, GET CHAR.
89AF  FE AC                CPI   ',' OR 200Q    ;IS IT A ','?
89B1  CA A1 89             JZ    SPACE          ;YES - SEND A SPACE
89B4  C3 DB 89             JMP   BMPTR          ;NO - SEND IT
89B7  CD 17 82     MODE1:  CALL  NUMB           ;NUMBER? (ALSO LOADS IT TO A)
89BA  DA DB 89             JC    BMPTR          ;YES - SEND IT AND BUMP PNTR'S
89BD  FE AE                CPI   256Q           ;DEC. PNT.?
89BF  CA DB 89             JZ    BMPTR
89C2  FE C5                CPI   305Q           ;E?
89C4  CA DB 89             JZ    BMPTR
89C7  FE AB                CPI   253Q           ;+?
89C9  CA D1 89             JZ    CHEKE
89CC  FE AD                CPI   255Q           ;-?
89CE  C2 A1 89             JNZ   SPACE          ;SEND A SPACE
89D1  47           CHEKE:  MOV   B,A            ;CHEK IF E PRECEDES +,-
89D2  2B                   DCX   H              ;BACK UP AND GET PRE-
89D3  7E                   MOV   A,M            ;CEDING CHARACTER
89D4  FE C5                CPI   305Q           ;IS IT E?
89D6  C2 A1 89             JNZ   SPACE          ;NO,+OR- WAS DELIMITTER
89D9  78                   MOV   A,B            ;YES,GET + OR -
89DA  23                   INX   H              ;RESTORE H,L
89DB  23           BMPTR:  INX   H              ;BUMP AND STORE PNTR'S
89DC  0D                   DCR   C
89DD  22 86 04             SHLD  HLINP
89E0  21 84 04             LXI   H,CREG
89E3  71                   MOV   M,C
89E4  C3 A3 89             JMP   IDONE          ;RESTORE REG'S AND RETURN
                   ;THIS ROUTINE WILL EVALUATE UNARY AND/OR
                   ;BINARY EXPRESIONS CALLED WITH H AND L
                   ;POINTING TO FIRST CHAR. OF EXP.,C CONTAINS
                   ;NUMBER OF CHAR'S LEFT IN LINE. RETURNS
                   ;D(H) AND E(L) POINTING TO THE ANSWER
                   ;THIS ROUTINE CALLS ITSELF RECURSIVELY
                   ;IN ORDER TO EVALUATE SUBSCRIPT
```

```
                        ;EXPRESIONS.  REG A,B DESTROYED
                        ;C,H,L ARE UPDATED
89E7    3E AD           EVAL:   MVI     A,255Q          ;IS IT UNARY -
89E9    BE                      CMP     M               ;Z=1 => YES
89EA    F5                      PUSH    PSW             ;Z=0 => NO
89EB    C2 F1 89                JNZ     ECAV
89EE    CD 05 86                CALL    ICP8            ;BUMP POINTER
89F1    CD DE 87        ECAV:   CALL    VALUE           ;GET PNTR. TO VALUE
89F4    E5                      PUSH    H               ;GET VALUE TO FREG1
89F5    21 7C 04                LXI     H,FREG1
89F8    CD C3 87                CALL    COPDH
89FB    EB                      XCHG
89FC    E1                      POP     H
89FD    F1                      POP     PSW             ;GET SIGN
89FE    C2 0C 8A                JNZ     DOL             ;SHALL WE NEGATE?
8A01    13                      INX     D               ;YES, POINT TO CHAR.
8A02    13                      INX     D
8A03    13                      INX     D
8A04    1A                      LDAX    D               ;AND LOAD TO A
8A05    17                      RAL                     ;ROTATE SIGN TO CY
8A06    3F                      CMC                     ;COMPLEMENT IT
8A07    1F                      RAR                     ;ROTATE BACK
8A08    12                      STAX    D               ;STORE AWAY
8A09    1B                      DCX     D               ;AND FIX PNTR.
8A0A    1B                      DCX     D
8A0B    1B                      DCX     D
8A0C    79              DOL:    MOV     A,C             ;IS THIS END OF LINE?
8A0D    B7                      ORA     A
8A0E    C8                      RZ                      ;YES-RETURN
8A0F    C5                      PUSH    B               ;SAVE C
8A10    3E 02                   MVI     A,2             ;NO SET UP TO CALL
8A12    CD 00 83                CALL    SYMSRT          ;SYMSRT AND CALL
8A15    C1                      POP     B               ;RESTORE C
8A16    3C                      INR     A               ;DELIMITER FOUND?
8A17    CA 42 8A                JZ      ER8             ;NO, ERROR
8A1A    D6 0A           EOK:    SUI     10              ;CHECK FOR EXPRESSION
8A1C    D8                      RC                      ;DELIMITER
8A1D    F5                      PUSH    PSW             ;SAVE OVERATION
8A1E    CD 05 86                CALL    ICP8            ;BUMP PNTR'S
8A21    B7                      ORA     A               ;CLEAR CY
8A22    E5              AGA:    PUSH    H               ;GET BYTES OF NUMBER
```

```
8A23  1A                  LDAX   D              ;AND PLACE ON STACK
8A24  6F                  MOV    L,A
8A25  13                  INX    D
8A26  1A                  LDAX   D
8A27  13                  INX    D
8A28  67                  MOV    H,A            ;2 BYTES TO H,L
8A29  E3                  XTHL                  ;XCHANGE, RESTORES H,L
8A2A  3F                  CMC
8A2B  DA 22 8A            JC     AGA            ;ANOTHER PASS?
8A2E  CD DE 87            CALL   VALUE          ;GET 2ND VALUE
8A31  79                  MOV    A,C            ;CHECK FOR END OF LINE
8A32  B7                  ORA    A              ;IF SO => WELL FORMED
8A33  CA 47 8A            JZ     WFOR
8A36  C5                  PUSH   B              ;SAVE C
8A37  3E 02               MVI    A,2            ;ELSE CALL SYMSRT TO
8A39  CD 00 83            CALL   SYMSRT         ;CHEK FOR EXP. DEL.
8A3C  C1                  POP    B              ;RECOVER IT
8A3D  FE 0A               CPI    10
8A3F  DA 47 8A            JC     WFOR           ;YES, WELL FORMED
8A42  3E 08         ER8:  MVI    A,8            ;ILL-FORMED EXP.
8A44  C3 C8 85            JMP    ERROR
8A47  C5            WFOR: PUSH   B              ;SAVE C, AND H,L
8A48  E5                  PUSH   H
8A49  21 80 04            LXI    H,FREG2        ;COPY 2ND VALUE TO
8A4C  CD C3 87            CALL   COPDH          ;FREG2
8A4F  D1                  POP    D              ;GET BYTES FROM STACK
8A50  C1                  POP    B
8A51  E1                  POP    H              ;INTO FREG1+2
8A52  22 7E 04            SHLD   FREG1+2
8A55  E1                  POP    H              ;AND NEXT 2 BYTES
8A56  22 7C 04            SHLD   FREG1          ;FROM STACK TO FREG1
8A59  EB                  XCHG
8A5A  F1                  POP    PSW            ;GET OPERATION
;THIS ROUTINE PERFORMS BINARY OPERATIONS ON OPERANDS IN FREG1 AND
;FREG2 B,C,H,L ARE LEFT UNDISTURBED. A IS DESTROYED
;D,E PNT TO RESULT
;OPERATIONS ARE SPECIFIED BY A REGISTER AS FOLLOWS:
;
;         A=0    =>          FREG1 * FREG2
;         A=1    =>          FREG1 / FREG2
;         A=2    =>          FREG1 + FREG2
```

```
                        ;          A=3     =>                FREG1 - FREG2
                        ;
                        ;IN CASE OF ARITHMETIC ERROR A MESSAGE IS SENT TO USER.
                        ;IF A CONTAINS ILLEGAL OPERATION REQUEST ERROR IS SENT TO USER
                        ;(ERROR 8) AND THE INTERPRETER IS ABORTED.
8A5B    C5              BINOP:  PUSH    B               ;SAVE REG'S
8A5C    E5                      PUSH    H
8A5D    21 7C 04                LXI     H,FREG1         ;SET UP PNTR'S TO
8A60    06 80                   MVI     B,FREG2 AND 377Q ;FREG'S AND SCR AREA
8A62    0E 66                   MVI     C,SCR AND 377Q  ;AND DO OPERATION
8A64    3D                      DCR     A
8A65    FA A7 8A                JM      FMULT           ;0,1=>* OR /
8A68    CA AD 8A                JZ      DIV             ;2,3=>+ OR -
8A6B    3D                      DCR     A
8A6C    CA 76 8A                JZ      ADDD
8A6F    3D                      DCR     A
8A70    CA A1 8A                JZ      SUBB
8A73    C3 42 8A                JMP     ER8             ;ILLEGAL OPER.
8A76    CD D0 97        ADDD:   CALL    LADD            ;DO ADDITION
8A79    54              ASBC:   MOV     D,H             ;FIX PNTR'S FOR RET.
8A7A    5D                      MOV     E,L
8A7B    B7              FPERR:  ORA     A               ;SET FLAGS
8A7C    CA 9E 8A                JZ      NFPER           ;NO ERROR
8A7F    D5                      PUSH    D               ;SAVE DE
8A80    F5                      PUSH    PSW             ;SAVE A
8A81    CD D3 82                CALL    WRIT            ;DUMP BUFFER
8A84    F1                      POP     PSW             ;GET A BACK
8A85    21 97 8A                LXI     H,WFPER         ;RETURN ADDRESS
8A88    E5                      PUSH    H               ;SAVE ON STACK
8A89    21 69 84                LXI     H,ODATA         ;MESSAGE TABLE
8A8C    17                      RAL                     ;UNDERFLOW?
8A8D    DA 4E 84                JC      FOR12           ;YES
8A90    17                      RAL                     ;OVERFLOW?
8A91    DA 4F 84                JC      FOR11           ;YES
8A94    C3 50 84                JMP     FOR10           ;NO - ITS ZERODIVIDE
8A97    21 69 84        WFPER:  LXI     H,ODATA         ;MESSAGE TABLE
8A9A    CD E9 85                CALL    ERLN            ;PRINT 'IN LINE --' (USE PART OF
8A9D    D1                      POP     D               ;ERROR RESTORE REG'S
8A9E    E1              NFPER:  POP     H
8A9F    C1                      POP     B
8AA0    C9                      RET
```

```
8AA1  CD D9 97  SUBB:   CALL  LSUB      ;DO SUBTRACTION
8AA4  C3 79 8A          JMP   ASBC
8AA7  CD D3 97  FMULT:  CALL  LMUL      ;DO MULT.
8AAA  C3 B0 8A          JMP   MDBC
8AAD  CD D6 97  DIV:    CALL  LDIV      ;DO DIV.
8AB0  54        MDBC:   MOV   D,H       ;AND FIX PNTR'S FOR RET.
8AB1  59                MOV   E,C
8AB2  C3 7B 8A          JMP   FPERR     ;CHECK FOR ERROR
                ;PRINT PROCESSOR
8AB5  2A 5B 04  PRI:    LHLD  CPNT
8AB8  23                INX   H         ;INCR. PAST KEYWORD
8AB9  23                INX   H
8ABA  23                INX   H
8ABB  CD 00 86          CALL  ICP7
8ABE  23                INX   H         ;BUMP PNTRS
8ABF  0D                DCR   C
8AC0  06 00             MVI   B,0       ;SET CHAR CNT
8AC2  C2 CE 8A          JNZ   PLOOP     ;CONTINUE IF MORE
8AC5  04                INR   B         ;NOTHING MORE, PAD A NULL
8AC6  3E 00             MVI   A,0
8AC8  CD B0 82          CALL  PAD
8ACB  C3 66 8B          JMP   PEND      ;WRITE IT AND CONTINUE
8ACE  7E        PLOOP:  MOV   A,M       ;GET CHARACTER
8ACF  FE A2             CPI   '"'+200Q  ;IS IT "?
8AD1  C2 F6 8A          JNZ   EXPRE     ;NO
8AD4  CD 00 86  QUOTE:  CALL  ICP7      ;GET CHARACTER TO A
8AD7  7E                MOV   A,M
8AD8  FE A2             CPI   '"'+200Q  ;IS IT "?
8ADA  CA E8 8A          JZ    QCHEK
8ADD  04        QOTOK:  INR   B         ;INCREMENT CNT
8ADE  50                MOV   D,B       ;SAVE IN D
8ADF  06 01             MVI   B,1       ;PAD ONCE
8AE1  CD B0 82          CALL  PAD
8AE4  42                MOV   B,D       ;RESTORE CNT
8AE5  C3 D4 8A          JMP   QUOTE     ;AGAIN
8AE8  23        QCHEK:  INX   H         ;BUMP PNTRS
8AE9  0D                DCR   C
8AEA  CA 66 8B          JZ    PEND      ;EOL
8AED  7E                MOV   A,M
8AEE  FE A2             CPI   '"'+200Q  ;ANOTHER "?
8AF0  CA DD 8A          JZ    QOTOK
```

```
8AF3  C3 27 8B          JMP     SCOLN
8AF6  CD 25 82  EXPRE:  CALL    ALPHA           ;IS IT A LETTER
8AF9  DA 0D 8B          JC      PRTIT           ;YES, EVALUATE AND PRINT
8AFC  CD 17 82          CALL    NUMB            ;IS IT A NUMB?
8AFF  DA 0D 8B          JC      PRTIT           ;YES, EVALUATE AND PRINT
8B02  7E                MOV     A,M
8B03  FE AE             CPI     '.'+200Q        ;IS IT A DECIMAL PNT?
8B05  CA 0D 8B          JZ      PRTIT           ;YES EVALUATE, PRINT
8B08  FE AD             CPI     '-'+200Q        ;IS IT A -?
8B0A  C2 27 8B          JNZ     SCOLN           ;NO, CHECK FOR ;
8B0D  C5        PRTIT:  PUSH    B               ;SAVE CNT
8B0E  CD E7 89          CALL    EVAL            ;EVALUATE EXPRESION
8B11  C5                PUSH    B               ;SAVE C,H,L
8B12  E5                PUSH    H
8B13  EB                XCHG                    ;DE TO HL
8B14  0E 66             MVI     C,SCR AND 377Q  ;SET UP, CONVERT
8B16  CD E5 97          CALL    CONV
8B19  E1                POP     H               ;RESTORE REG'S
8B1A  C1                POP     B
8B1B  79                MOV     A,C
8B1C  C1                POP     B
8B1D  4F                MOV     C,A
8B1E  B7                ORA     A               ;CHECK EOL
8B1F  CA 66 8B          JZ      PEND
8B22  3E 0B             MVI     A,11            ;UPDATE CNTR
8B24  80                ADD     B
8B25  47                MOV     B,A
8B26  7E                MOV     A,M             ;GET CHAR.
8B27  FE BB     SCOLN:  CPI     ';'+200Q        ;IS IT ;?
8B29  CA 59 8B          JZ      SONWD           ;YES
8B2C  FE AC             CPI     ','+200Q        ;IS IT ,?
8B2E  C2 44 87          JNZ     ER6             ;NO-UNEXPECTED CHAR.
8B31  AF                XRA     A               ;ZERO A
8B32  C6 0D     ADFLD:  ADI     13              ;ADD FIELD LENGTH
8B34  B8                CMP     B               ;COMPARE TO CNT
8B35  CA 3B 8B          JZ      $+6
8B38  D2 4D 8B          JNC     FLDFD
8B3B  FE 34             CPI     52              ;LAST FLD?
8B3D  C2 32 8B          JNZ     ADFLD
8B40  CD D3 82          CALL    WRIT            ;YES-WRITE LINE
8B43  06 00             MVI     B,0             ;RESET CNT
```

```
8B45  23          ONWD:   INX   H              ;BUMP PNTRS
8B46  0D                  DCR   C
8B47  CA 66 8B            JZ    PEND
8B4A  C3 CE 8A            JMP   PLOOP
8B4D  90          FLDFD:  SUB   B              ;FOUND FIELD
8B4E  50                  MOV   D,B            ;DETERMIN   OF SPACES TO PAD
8B4F  5F                  MOV   E,A            ;SET UP TO CALL PAD
8B50  47                  MOV   B,A
8B51  3E A0               MVI   A,240Q
8B53  CD B0 82            CALL  PAD            ;PAD SPACES
8B56  7A                  MOV   A,D
8B57  83                  ADD   E              ;NEW CNT
8B58  47                  MOV   B,A            ;SAVE IN B
8B59  23          SONWD:  INX   H              ;CHECK EOL
8B5A  0D                  DCR   C
8B5B  C2 CE 8A            JNZ   PLOOP
8B5E  16 01               MVI   D,1            ;SUPPRESS CR/LF
8B60  CD D5 82            CALL  WRIT1
8B63  C3 69 8B            JMP   $+6
8B66  CD D3 82    PEND:   CALL  WRIT           ;DUMP BUFFER, CONTINUE
8B69  C3 AC 8B            JMP   IEND
                  ;INPUT PROCESSOR - READS VALUES FROM TTY
                  ;THEY MUST BE DELIMITED BY COMMAS ONLY
8B6C  79          INPUT:  MOV   A,C            ;IN CASE OF ERROR
8B6D  32 5F 04            STA   PL6            ;SAVE
8B70  2A 5B 04    INPER:  LHLD  CPNT           ;INPUT LINE (V-STRING) PNTR
8B73  23                  INX   H              ;ADJUST PNTR'S
8B74  23                  INX   H
8B75  23                  INX   H
8B76  CD 00 86            CALL  ICP7
8B79  CD 00 86            CALL  ICP7
8B7C  C5          PRMPT:  PUSH  B              ;SAVE PNTR'S
8B7D  E5                  PUSH  H
8B7E  06 01               MVI   B,1            ;SEND PROMPT
8B80  3E 3A               MVI   A,':'
8B82  50                  MOV   D,B            ;TO SUPPRESS CR/LF
8B83  CD B0 82            CALL  PAD            ;PAD IT
8B86  CD D5 82            CALL  WRIT1          ;WRITE IT
8B89  21 01 04            LXI   H,IBUF         ;ADD. OF INPUT BUFFER
8B8C  CD D2 83            CALL  TTYIN          ;READ A LINE
8B8F  EB                  XCHG                 ;ADD. OF K-STRING TO DE
```

```
8B90  E1              POP   H               ;ADD. OF V-STRING
8B91  C1              POP   B               ;V-STRING CNT TO C
8B92  47              MOV   B,A             ;K-STRING CNT TO B
8B93  CD B2 8B        CALL  STRIN           ;TRANSFER CONSTANTS TO VARIBLES
8B96  CA A9 8B        JZ    INPOK           ;NO ERROR
8B99  21 69 84        LXI   H,ODATA         ;SEND ERROR MESSAGE
8B9C  CD 51 84        CALL  FORM9
8B9F  CD D3 82        CALL  WRIT
8BA2  3A 5F 04        LDA   PL6             ;GET V-STRING CNT
8BA5  4F              MOV   C,A
8BA6  C3 70 8B        JMP   INPER           ;START AGAIN
8BA9  DA 7C 8B  INPOK: JC    PRMPT          ;NEED MORE CONSTANTS
8BAC  2A 56 04  IEND:  LHLD  KFPNT          ;ALL OK - GET NEW PNTR.
8BAF  C3 A4 86        JMP   ILOOP           ;CONTINUE
                ;THIS ROUTINE TRANSFERS THE FLOATING POINT VALUES
                ;OF AN ASCII STRING OF CONSTANTS TO THE LOCATIONS
                ;SPECIFIED BY AN ASCII STRING OF VARIBLES
                ;POINTER AND LINE CNT OF VAR. STRING ARE IN HL,C
                ;POINTER AND LINE CNT OF CONST. STRING ARE IN DE,B
                ;ON RETURN:
                ;          Z=0 AND CY=0   ALL OK
                ;          Z=0 AND CY=1   NEED MORE CONSTANTS
                ;          Z=1            ERROR IN CONVERSION
                ;ALL POINTERS AND LINE CNT'S ARE RETURNED UPDATED
8BB2  79        STRIN: MOV   A,C             ;GET V-STRING CNT
8BB3  B7               ORA   A               ;TEST FOR EOL
8BB4  C8               RZ                    ;DONE, CY=0 => ALL OK
8BB5  7E               MOV   A,M             ;GET CHAR.
8BB6  FE AC            CPI   ',' OR 200Q     ;IS IT A ,?
8BB8  C2 CO 8B         JNZ   STOKV           ;IT'S NOT A ,
8BBB  23               INX   H               ;COMMA, BUMP PNTR'S
8BBC  OD               DCR   C
8BBD  CA E8 8B         JZ    ERRET           ;POSSIBLE ERROR (IF EOL)
8BC0  78        STOKV: MOV   A,B             ;GET K-STRING LENGTH
8BC1  B7               ORA   A               ;TEST FOR EOL
8BC2  37               STC                   ;IN CASE IT'S EOL
8BC3  C8               RZ                    ;RET, CY=1 =@ NEED MORE CONSTANTS
8BC4  1A               LDAX  D               ;GET CHAR
8BC5  FE AC            CPI   ',' OR 200Q     ;TEST FOR ,
8BC7  C2 CF 8B         JNZ   STOKK           ;NOT A ',' - READY TO GO
8BCA  13               INX   D               ;BUMP PNTR'S
```

```
8BCB  05              DCR   B
8BCC  CA E8 8B        JZ    ERRET       ;POSSIBLE ERROR (IF EOL)
8BCF  C5        STOKK: PUSH  B           ;SAVE K-STRING CNT
8BD0  D5              PUSH  D           ;SAVE K-STRING PNTR
8BD1  CD 8D 88        CALL  VAR         ;ADD. TO VARIBLE TO DE
8BD4  EB              XCHG              ;VAR. ADD TO H,L
8BD5  22 8A 04        SHLD  VARAD       ;SAVE
8BD8  E1              POP   H           ;ADDRESS OF K-STRING
8BD9  79              MOV   A,C         ;V-STRING CNT TO A
8BDA  C1              POP   B           ;K-STRING CNT TO B
8BDB  48              MOV   C,B         ;K-STRING CNT TO C
8BDC  F5              PUSH  PSW         ;SAVE V-STRING CNT
8BDD  D5              PUSH  D           ;SAVE V-STRING ADD.
8BDE  3E 00           MVI   A,0         ;A=0 =@ DATA FROM TTY
8BE0  CD 6E 88        CALL  RDKON       ;GET CONSTANT TO GREG
8BE3  D2 EB 8B        JNC   STNER
8BE6  E1              POP   H           ;EMPTY STACK
8BE7  E1              POP   H
8BE8  AF        ERRET: XRA  A           ;ERROR
8BE9  3C              INR   A
8BEA  C9              RET
8BEB  E5        STNER: PUSH  H          ;SAVE K-STRING PNTR.
8BEC  2A 8A 04        LHLD  VARAD       ;GET VAR. ADD
8BEF  11 77 04        LXI   D,GREG      ;ADD. TO CONST.
8BF2  CD C3 87        CALL  COPDH       ;COPY IT TO VARIABLE LOC.
8BF5  D1              POP   D           ;K-STING PNTR. TO DE
8BF6  41              MOV   B,C         ;K-STRING LENGTH TO B
8BF7  E1              POP   H           ;V-STRING PNTR. TO HL
8BF8  F1              POP   PSW         ;V-STRING LENGTH TO C
8BF9  4F              MOV   C,A
8BFA  C3 B2 8B        JMP   STRIN       ;LOOP
                ;LET STMT. PROCESSOR
8BFD  2A 5B 04  LET:  LHLD  CPNT        ;GET PNTR.
8C00  23              INX   H           ;FIX PNTR.
8C01  23              INX   H
8C02  23              INX   H
8C03  79              MOV   A,C         ;CHECK FOR EOL
8C04  B7              ORA   A
8C05  C2 0D 8C        JNZ   LOK
8C08  3E 07     ER7:  MVI   A,7
8C0A  C3 C8 85        JMP   ERROR
```

```
8C0D  CD 8D 88        LOK:    CALL    VAR             ;GET ADDRESS TO VAR.
8C10  DA 52 8C                JC      SAVV            ;IT'S A VARIABLE
8C13  3E 03                   MVI     A,3             ;NO-CHEK FOR FUNC.
8C15  CD 00 83                CALL    SYMSRT
8C18  FE FF                   CPI     377Q
8C1A  CA 42 8A                JZ      ER8             ;DON'T KNOW WHAT IT IS
8C1D  3D                      DCR     A
8C1E  C2 98 8C                JNZ     ER10            ;ILLEGAL USE OF FUNC.
8C21  23                      INX     H               ;IT'S PUT,UPDATE H,L
8C22  23                      INX     H
8C23  23                      INX     H
8C24  79                      MOV     A,C             ;EOL CHK
8C25  B7                      ORA     A
8C26  CA 42 8A                JZ      ER8
8C29  7E                      MOV     A,M             ;CHEK FOR (
8C2A  FE A8                   CPI     250Q
8C2C  C2 42 8A                JNZ     ER8
8C2F  CD 05 86                CALL    ICP8            ;BUMP PNTRS
8C32  CD E7 89                CALL    EVAL            ;EVALUATE AND FIX
8C35  E5                      PUSH    H               ;SAVE H,L
8C36  21 7C 04                LXI     H,FREG1
8C39  CD C3 87                CALL    COPDH           ;COPY IT
8C3C  EB                      XCHG
8C3D  E1                      POP     H
8C3E  CD 35 89                CALL    FIX
8C41  13                      INX     D
8C42  13                      INX     D
8C43  13                      INX     D
8C44  1A                      LDAX    D               ;GET LOWEST BYTE
8C45  F5                      PUSH    PSW             ;PORT = IS SAVED
8C46  7E                      MOV     A,M
8C47  FE A9                   CPI     251Q            ;CHECK FOR )
8C49  C2 42 8A                JNZ     ER8
8C4C  CD 05 86                CALL    ICP8            ;BUMP PNTR'S
8C4F  16 FF                   MVI     D,377Q
8C51  5A                      MOV     E,D
8C52  D5              SAVV:   PUSH    D               ;KEEP ADDRESS
8C53  7E                      MOV     A,M             ;CHEK FOR =
8C54  FE BD                   CPI     275Q
8C56  C2 42 8A                JNZ     ER8
8C59  CD 05 86                CALL    ICP8            ;BUMP PNTRS
```

```
8C5C  CD E7 89            CALL   EVAL          ;EVALUATE EXPRESSION
8C5F  E1                  POP    H             ;GET ADDRESS
8C60  CD A6 82            CALL   CHK1
8C63  DA 6C 8C            JC     PTFIN         ;IT WAS A PUT
8C66  CD C3 87            CALL   COPDH         ;COPY TO ADDRESS
8C69  C3 AC 8B            JMP    IEND          ;CONTINUE
8C6C  21 7C 04  PTFIN:    LXI    H,FREG1       ;COPY VALUE TO FREG1
8C6F  CD C3 87            CALL   COPDH
8C72  EB                  XCHG
8C73  CD 35 89            CALL   FIX           ;FIX THE VALUE
8C76  13                  INX    D
8C77  13                  INX    D
8C78  13                  INX    D
8C79  1A                  LDAX   D
8C7A  4F                  MOV    C,A           ;SAVE IN C
8C7B  21 93 8C            LXI    H,PINST       ;ADD OF BYTES TO GO TO
8C7E  11 77 04            LXI    D,GREG        ;RAM AT GREG
8C81  06 05               MVI    B,5           ;BYTE CNT
8C83  7E        PRI1:     MOV    A,M           ;STORE PROG. SEG. IN
8C84  12                  STAX   D             ;RAM
8C85  23                  INX    H
8C86  13                  INX    D
8C87  05                  DCR    B
8C88  C2 83 8C            JNZ    PRI1
8C8B  F1                  POP    PSW           ;GET PORT =
8C8C  21 78 04            LXI    H,GREG+1
8C8F  77                  MOV    M,A           ;STORE
8C90  79                  MOV    A,C           ;GET DATA OUT TO A
8C91  2B                  DCX    H             ;TRANSFER
8C92  E9                  PCHL
8C93  D3 00     PINST:    OUT    0             ;RAM INSTRUCTIONS
8C95  C3 AC 8B            JMP    IEND
8C98  3E 10     ER10:     MVI    A,10H
8C9A  C3 C8 85            JMP    ERROR
                ;IF STMT. PROCESSOR
8C9D  2A 5B 04  IFRT:     LHLD   CPNT          ;GET PNTR., ADJUST
8CA0  23                  INX    H
8CA1  0C                  INR    C             ;CHECK EOL
8CA2  CD 00 86            CALL   ICP7
8CA5  CD E7 89            CALL   EVAL          ;EVALUATE EXPRESSION
8CA8  79                  MOV    A,C
```

```
8CA9  B7                ORA   A          ;CHECK EOL
8CAA  CA 08 8C          JZ    ER7
8CAD  E5          IAGA: PUSH  H          ;SAVE H,L, PUT VALUE ON STK
8CAE  1A                LDAX  D
8CAF  13                INX   D
8CB0  6F                MOV   L,A
8CB1  1A                LDAX  D
8CB2  13                INX   D
8CB3  67                MOV   H,A
8CB4  E3                XTHL
8CB5  3F                CMC                ;RESTORE H,L
8CB6  DA AD 8C          JC    IAGA         ;ANOTHER PASS?
8CB9  3E 02             MVI   A,2
8CBB  CD 00 83          CALL  SYMSRT       ;CHEK TYPE OF RELATION
8CBE  FE 04             CPI   4            ;WAS IT LEGAL?
8CC0  DA C8 8C          JC    III
8CC3  3E 14       ER14: MVI   A,14H
8CC5  C3 C8 85          JMP   ERROR
8CC8  FE 02       III:  CPI   2            ;WAS IT A ,?
8CCA  CA C3 8C          JZ    ER14
8CCD  3C                INR   A            ;ALL OK, INC,SAVE
8CCE  F5                PUSH  PSW
8CCF  0C                INR   C
8CD0  CD 00 86          CALL  ICP7         ;BUMP PNTRS
8CD3  3E 02             MVI   A,2          ;CALL SYMSRT
8CD5  CD 00 83          CALL  SYMSRT
8CD8  FE FF             CPI   377Q         ;FOUND ANYTHING?
8CDA  CA F5 8C          JZ    RELAT        ;DONE
8CDD  FE 02             CPI   2
8CDF  CA C3 8C          JZ    ER14         ;IT WAS A ,
8CE2  FE 04             CPI   4
8CE4  D2 C3 8C          JNC   ER14         ;NOT LEGAL
8CE7  3C                INR   A
8CE8  47                MOV   B,A
8CE9  0C                INR   C
8CEA  CD 00 86          CALL  ICP7
8CED  F1                POP   PSW          ;GET SECOND RELATION
8CEE  80                ADD   B            ;ADD THEM
8CEF  F5                PUSH  PSW          ;AND SAVE
8CF0  FE 08             CPI   10Q          ;TEST FOR ==
8CF2  CA C3 8C          JZ    ER14
```

```
                        ;RELATION IS STORED ON TOP OF STACK (PUSH PSW) ACCORDING TO
                        ;THE FOLLOWING
                        ;
                        ;           1 =>    <
                        ;           2 =>    >
                        ;           3 =>    <>
                        ;           4 =>    =
                        ;           5 =>    <=
                        ;           6 =>    >=
                        ;
8CF5    CD E7 89        RELAT:  CALL    EVAL            ;EVALUATE
8CF8    E5                      PUSH    H               ;SAVE H,L
8CF9    21 80 04                LXI     H,FREG2         ;COPY TO FREG2
8CFC    CD C3 87                CALL    COPDH
8CFF    E1                      POP     H               ;GET H,L
8D00    F1                      POP     PSW             ;AND RELATION
8D01    E3                      XTHL                    ;GET 2ND 2 BYTES
8D02    22 7E 04                SHLD    FREG1+2         ;STORE
8D05    E1                      POP     H               ;GET 1ST 2 BYTES,STORE
8D06    E3                      XTHL
8D07    22 7C 04                SHLD    FREG1
8D0A    C5                      PUSH    B
8D0B    F5                      PUSH    PSW             ;SAVE A,B,C
8D0C    CD 43 8D                CALL    FCOMP           ;COMPARE NUMBERS
8D0F    57                      MOV     D,A             ;SAVE RESULT IN D
8D10    F1                      POP     PSW             ;GET RELATION,B,C
8D11    C1                      POP     B
8D12    BA                      CMP     D               ;SAME?
8D13    CA 36 8D                JZ      TRUE            ;YES
8D16    D6 04                   SUI     4
8D18    F2 28 8D                JP      NOT3            ;NOT RELATION 3
8D1B    3C                      INR     A               ;IS IT RELATION 3?
8D1C    C2 32 8D                JNZ     FALSE           ;NO, ITS FALSE
8D1F    3E 04                   MVI     A,4             ;IT IS, CHECK FOR INEQUALITY
8D21    BA                      CMP     D
8D22    C2 36 8D                JNZ     TRUE
8D25    C3 32 8D                JMP     FALSE
8D28    BA              NOT3:   CMP     D               ;RELATION 5,6 TRUE?
8D29    CA 36 8D                JZ      TRUE            ;YES
8D2C    3E 04                   MVI     A,4             ;IT WAS, CHECK FOR EQUALITY
8D2E    BA                      CMP     D
```

```
8D2F  CA 36 8D                    JZ      TRUE
8D32  E1          FALSE:          POP     H                 ;CONTINUE
8D33  C3 AC 8B                    JMP     IEND
8D36  E1          TRUE:           POP     H
8D37  06 04                       MVI     B,4
8D39  CD 00 86    THEN:           CALL    ICP7              ;INCREMENT PAST THEN
8D3C  05                          DCR     B
8D3D  C2 39 8D                    JNZ     THEN
8D40  C3 21 87                    JMP     GTRA              ;TRANSFER TO GOTO
                  ;ROUTINE FCOMP COMPARES 2 FLOATING POINT #'S.  THEY ARE ASSUMED
                  ;TO BE IN FREG1 AND FREG2.
                  ;ALL REGISTERS ARE DESTROYED.
                  ;THE VALUE RETURNED IN REG A IS RESULT OF COMPARISON.
                  ;RESULTS ARE AS FOLLOWS:
                  ;
                  ;         A=1     =>      FREG1 < FREG2
                  ;         A=2     =>      FREG1 > FREG2
                  ;         A=4     =>      FREG1 = FREG2
                  ;
8D43  21 7F 04    FCOMP:          LXI     H,FREG1+3         ;PNTS TO CHAR OF 1ST
8D46  11 83 04                    LXI     D,FREG2+3         ;PNTS TO CHAR OF 2ND
8D49  7E                          MOV     A,M               ;GET  1 CHAR
8D4A  06 80                       MVI     B,200Q            ;MASK TO B
8D4C  A0                          ANA     B                 ;GET SIGN,  1
8D4D  4F                          MOV     C,A               ;SAVE IN C
8D4E  1A                          LDAX    D                 ;GET CHAR  2
8D4F  A0                          ANA     B                 ;GET SIGN  2
8D50  A9                          XRA     C
8D51  CA 5B 8D                    JZ      SINEQ             ;SAME SIGNS
8D54  79                          MOV     A,C               ;OPPISITE SIGNS,GET  1 SIGN
8D55  17                          RAL                       ;ROTATE TO CY
8D56  3E 01                       MVI     A,1
8D58  D8                          RC                        ;FREG1 < FREG2 => A=1
8D59  3C                          INR     A                 ;ELSE FREG1 > FREG2
8D5A  C9                          RET                       ;AND A=2
8D5B  C5          SINEQ:          PUSH    B                 ;SAVE SIGN
8D5C  2B                          DCX     H                 ;PNTR TO  1 IN H,L
8D5D  2B                          DCX     H
8D5E  2B                          DCX     H
8D5F  43                          MOV     B,E               ;PNTR TO  2 IN B
8D60  05                          DCR     B
```

```
8D61  05                  DCR   B
8D62  05                  DCR   B
8D63  CD DF 97            CALL  LMCM          ;COMPARE MAGNITUDES
                    ;AT THIS POINT Z=1 => =, CY=1 => 1<2
8D66  C1                  POP   B             ;GET SIGN BACK
8D67  C2 6D 8D            JNZ   $+6
8D6A  3E 04               MVI   A,4           ;EQUAL => A=4
8D6C  C9                  RET
8D6D  79                  MOV   A,C           ;GET SIGN TO A
8D6E  3C                  INR   A             ;SET SIGN BIT
8D6F  3E 01               MVI   A,1
8D71  FA 77 8D            JM    $+6.          ;SIGN IS NEGATIVE
8D74  D8                  RC                  ;SIGN=+ AND ABS(FREG1)<ABS(FREG2)
8D75  3C                  INR   A             ;ABS(FREG1)>ABS(FREG2)
8D76  C9                  RET
8D77  D0                  RNC                 ;SIGN=- AND ABS(FREG1)>ABS(FREG2)
8D78  3C                  INR   A             ;ABS(FREG1)<ABS(FREG2)
8D79  C9                  RET
                    ;CALL PROCESSOR
8D7A  21 AC 8B      CALLP:LXI   H,IEND        ;INIT RETURN ADDRESS
8D7D  E5                  PUSH  H
8D7E  2A 5B 04            LHLD  CPNT          ;INIT POINTERS
8D81  23                  INX   H
8D82  23                  INX   H
8D83  23                  INX   H
8D84  CD 00 86            CALL  ICP7
8D87  7E                  MOV   A,M           ;GET CHAR
8D88  FE A8               CPI   '('+200Q      ;IS IT A (?
8D8A  C2 08 8C            JNZ   ER7           ;BAD
8D8D  CD 00 86            CALL  ICP7          ;BUMP PNTRS
8D90  CD 2D 82            CALL  CVB           ;GET SUB
8D93  85                  ADD   L             ;UPDATA H,L
8D94  6F                  MOV   L,A
8D95  3E 00               MVI   A,0
8D97  8C                  ADC   H
8D98  67                  MOV   H,A           ;D NOW CONTAINS SUB
8D99  E5                  PUSH  H             ;SAVE HL
8D9A  2A 02 01            LHLD  SUBAD         ;GET START OF SUB TABLE
8D9D  7E            NUSUB:MOV   A,M           ;GET ENTRY
8D9E  BA                  CMP   D             ;COMPARE
8D9F  CA AE 8D            JZ    FNDSB         ;FOUND IT
```

```
8DA2  23           INX   H          ;PNT TO NEXT ENTRY
8DA3  23           INX   H
8DA4  23           INX   H
8DA5  3C           INR   A          ;CHECK TO SEE IF LAST WAS 377Q
8DA6  C2 9D 8D     JNZ   NUSUB
8DA9  3E 15        MVI   A,15H      ;ER 15 - NO SUB BY THIS =
8DAB  C3 C8 85     JMP   ERROR
8DAE  23     FNDSB: INX   H          ;FOUND IT,GET STARTING ADD.
8DAF  5E           MOV   E,M
8DB0  23           INX   H
8DB1  66           MOV   H,M
8DB2  6B           MOV   L,E        ;AND SAVE IT
8DB3  22 5D 04     SHLD  SBSAV
8DB6  2A 59 04     LHLD  NXTSP      ;INIT MEMORY SCRATCH AREA
8DB9  22 88 04     SHLD  MESCR
8DBC  E1           POP   H
8DBD  7E     PARLP: MOV   A,M        ;GET SOURCE PNTR BACK
8DBE  FE A9        CPI   ')'+200Q   ;GET CHAR
8DC0  CA EC 8D     JZ    CLSUB      ;IS IT )?
8DC3  FE AC        CPI   ','+200Q   ;YES - GO CALL SUB
8DC5  C2 44 87     JNZ   ER6        ;DO WE HAVE A ,?
8DC8  CD 00 86     CALL  ICP7       ;UEXPECTED CHARACTER
8DCB  CD 8D 88     CALL  VAR        ;BUMP PNTRS
8DCE  D2 D5 8D     JNC   PREXP      ;DO WE HAVE A VARIABLE
8DD1  D5           PUSH  D          ;NO
8DD2  C3 BD 8D     JMP   PARLP      ;YES - SAVE ADDRESS
8DD5  CD E7 89 PREXP: CALL  EVAL     ;CONTINUE
8DD8  E5           PUSH  H          ;EVALUATE EXPRESSION
8DD9  2A 88 04     LHLD  MESCR      ;SAVE H,L
8DDC  CD C3 87     CALL  COPDH      ;GET SCRATCH AREA
8DDF  D1           POP   D          ;AND COPY TO IT
8DE0  E5           PUSH  H          ;HL TO DE
8DE1  23           INX   H          ;SAVE ADDRESS
8DE2  23           INX   H          ;UPDATE MESCR
8DE3  23           INX   H
8DE4  23           INX   H
8DE5  22 88 04     SHLD  MESCR      ;SAVE IT
8DE8  EB           XCHG             ;GET H,L BACK
8DE9  C3 BD 8D     JMP   PARLP      ;CONTINUE
8DEC  2A 5D 04 CLSUB: LHLD  SBSAV    ;START OF ROUTINE
8DEF  E9           PCHL             ;TRANSFER
```

```
                        ;GOSUB PROCESSOR
8DF0    21 A4 86    GOSUB:  LXI     H,ILOOP     ;FOR RETURN STMT.
8DF3    E5              PUSH    H               ;TO STACK
8DF4    2A 56 04        LHLD    KFPNT           ;PNTR. TO NEXT STMT.
8DF7    E5              PUSH    H               ;SAVE ON STACK
8DF8    2A 59 04        LHLD    NXTSP           ;CHECK MEMORY
8DFB    CD 1D 85        CALL    MEMFUL
8DFE    2A 5B 04        LHLD    CPNT            ;GET CHAR. PNTR
8E01    23              INX     H
8E02    C3 1B 87        JMP     GSENT           ;PART OF GOTO TO FINISH
                        ;RETURN STMT. PROCESSOR
8E05    E1          RETRN:  POP     H           ;GET RETURN ADD. FROM STACK
8E06    C9              RET                     ;CONTINUE
                        ;FOR STATEMENT PROCESSOR
8E07    2A 5B 04    FOR:    LHLD    CPNT        ;FIX PNTRS
8E0A    0C              INR     C
8E0B    23              INX     H
8E0C    23              INX     H
8E0D    CD 00 86        CALL    ICP7
8E10    CD 25 82        CALL    ALPHA           ;LETTER?
8E13    D2 B5 8F        JNC     ER21            ;NO
8E16    46              MOV     B,M             ;GET IT TO B
8E17    CD 00 86        CALL    ICP7            ;BUMP PNTR'S
8E1A    51              MOV     D,C             ;SAVE C
8E1B    0E 00           MVI     C,0             ;INIT C TO 0
8E1D    CD 17 82        CALL    NUMB            ;NUMBER?
8E20    D2 29 8E        JNC     $+9             ;NO
8E23    4E              MOV     C,M             ;YES, GET IT
8E24    23              INX     H               ;BUMP PNTR'S
8E25    15              DCR     D
8E26    CA 08 8C        JZ      ER7             ;PREMATURE EOL
8E29    E5              PUSH    H               ;SAVE H,L
8E2A    CD 17 86        CALL    FSYM            ;GET VAR. LOCATION
8E2D    E3              XTHL                    ;PUT ON STACK, GET H,L
8E2E    59              MOV     E,C             ;VARIABLE TO D,E
8E2F    4A              MOV     C,D             ;RESTORE C
8E30    50              MOV     D,B
8E31    EB              XCHG                    ;SAVE VAR NAME
8E32    22 8C 04        SHLD    VNAME
8E35    EB              XCHG                    ;RESTORE H,L
8E36    7E              MOV     A,M             ;LOOK FOR =
```

```
8E37  FE BD          CPI   '=' OR 200Q
8E39  C2 9C 8F       JNZ   ER16
8E3C  CD 00 86       CALL  ICP7          ;BUMP PNTR'S
8E3F  CD E7 89       CALL  EVAL          ;EVALUATE EXPRESSION
8E42  E3             XTHL                ;VARIABLE LOCATION
8E43  CD C3 87       CALL  COPDH         ;WRITE VALUE
8E46  22 8E 04       SHLD  VLOC          ;SAVE PNTR TO VARIABLE LOCATION
8E49  E1             POP   H             ;GET H,L BACK
8E4A  79             MOV   A,C           ;CHECK EOL
8E4B  B7             ORA   A
8E4C  CA 08 8C       JZ    ER7
8E4F  3E 02          MVI   A,2           ;CHECK FOR 'TO'
8E51  CD 00 83       CALL  SYMSRT
8E54  FE 07          CPI   7
8E56  C2 A1 8F       JNZ   ER17
8E59  23             INX   H             ;BUMB PNTR'S
8E5A  23             INX   H
8E5B  79             MOV   A,C           ;CHECK EOL
8E5C  B7             ORA   A
8E5D  CA 08 8C       JZ    ER7
8E60  CD E7 89       CALL  EVAL          ;EVALUATE LIMIT
8E63  E5             PUSH  H             ;SAVE H,L
8E64  21 90 04       LXI   H,FLIMT       ;SAVE LIMIT VALUE
8E67  CD C3 87       CALL  COPDH
8E6A  79             MOV   A,C           ;CHECK EOL
8E6B  B7             ORA   A
8E6C  C2 76 8E       JNZ   STP
8E6F  11 F5 8E       LXI   D,FONE        ;DEFAULT STEP=1
8E72  E1             POP   H             ;RESTORE H,L
8E73  C3 8B 8E       JMP   FBILD
8E76  E1             POP   H             ;GET H,L
8E77  3E 02          MVI   A,2           ;LOOK FOR 'STEP'
8E79  CD 00 83       CALL  SYMSRT
8E7C  FE 08          CPI   8
8E7E  C2 A1 8F       JNZ   ER17
8E81  23             INX   H             ;FIX H,L
8E82  23             INX   H
8E83  23             INX   H
8E84  0C             INR   C             ;CHECK EOL
8E85  CD 00 86       CALL  ICP7
8E88  CD E7 89       CALL  EVAL          ;GET STEP SIZE
```

```
STP:
```

```
                        ;AT THIS POINT:
                        ;VARIABLE NAME IS IN LOCATION VNAME
                        ;VARIABLE ADDRESS IS IN LOCATION VLOC
                        ;VARIBLE HAS BEEN INITIALIZED
                        ;LIMIT IS IN 4 BYTE LOCATION FLIMT
                        ;STEP IS POINTED TO BY D,E
                        ;H,L,C ARE POINTER, COUNTER AS USUAL
8E8B  D5                FBILD:  PUSH    D               ;SAVE PNTR TO STEP
8E8C  2A 8C 04                  LHLD    VNAME           ;GET VARIABLE NAME
8E8F  3E 3F                     MVI     A,77Q           ;MASK
8E91  A4                        ANA     H               ;MASK OFF TOP 2 BITS
8E92  47                        MOV     B,A             ;SET UP TO CALL FSYM
8E93  4D                        MOV     C,L
8E94  CD 17 86                  CALL    FSYM            ;FIND ENTRY
8E97  DA AD 8E                  JC      FEXST           ;IT WAS THERE
8E9A  E5                        PUSH    H               ;IT WASN'T, SAVE H,L
8E9B  2A 59 04                  LHLD    NXTSP           ;UPDATE NXTSP
8E9E  3E 08                     MVI     A,8             ;ADD 8 TO H,L
8EA0  85                        ADD     L
8EA1  6F                        MOV     L,A
8EA2  3E 00                     MVI     A,0
8EA4  8C                        ADC     H
8EA5  67                        MOV     H,A
8EA6  22 59 04                  SHLD    NXTSP           ;NEW VALUE OF NXTSP
8EA9  CD 1D 85                  CALL    MEMFUL          ;CHECK MEMORY
8EAC  E1                        POP     H               ;GET ADD. IN DATA BLOCK
8EAD  D1                FEXST:  POP     D               ;ADDRESS OF STEP SIZE
8EAE  CD C3 87                  CALL    COPDH           ;STORE IT
8EB1  23                        INX     H               ;PNT TO WHERE VAR. PNTR GOES
8EB2  23                        INX     H
8EB3  23                        INX     H
8EB4  23                        INX     H
8EB5  3A 8E 04                  LDA     VLOC            ;FIRST BYTE
8EB8  77                        MOV     M,A             ;STORE IT
8EB9  23                        INX     H
8EBA  3A 8F 04                  LDA     VLOC+1          ;SECOND BYTE
8EBD  77                        MOV     M,A
8EBE  23                        INX     H               ;PNT TO WHERE LIMIT GOES
8EBF  11 90 04                  LXI     D,FLIMT         ;WHERE IT IS NOW
8EC2  CD C3 87                  CALL    COPDH           ;COPY IT
8EC5  23                        INX     H               ;PNT TO WHERE KFPNT GOES
```

```
8EC6    23                      INX     H
8EC7    23                      INX     H
8EC8    23                      INX     H
8EC9    3A 56 04                LDA     KFPNT           ;1ST BYTE
8ECC    77                      MOV     M,A
8ECD    23                      INX     H
8ECE    3A 57 04                LDA     KFPNT+1         ;2ND BYTE
8ED1    77                      MOV     M,A
                ;PUT CURRENT VNAME ON NESTING STACK
8ED2    21 00 00                LXI     H,0             ;GET STACK-POINTER
8ED5    39                      DAD     SP
8ED6    22 8E 04                SHLD    VLOC            ;SAVE IT
8ED9    2A 94 04                LHLD    NEST            ;GET NEST SP
8EDC    7D                      MOV     A,L             ;COMPARE WITH STACK LIMIT
8EDD    FE 96                   CPI     TOPNS AND 377Q  ;NEED ONLY COMPARE PAGE LOCATION
8EDF    CA A6 8F                JZ      ER18            ;FOR'S NEXTED TOO DEEPLY
8EE2    F9              NSTOK:  SPHL                    ;LOAD NEW SP
8EE3    EB                      XCHG                    ;SAVE NEST SP
8EE4    2A 8C 04                LHLD    VNAME           ;GET INDEX NAME
8EE7    E5                      PUSH    H               ;SAVE IT
8EE8    1B                      DCX     D               ;UPDATE NEST SP
8EE9    1B                      DCX     D
8EEA    EB                      XCHG                    ;SAVE IT
8EEB    22 94 04                SHLD    NEST
8EEE    2A 8E 04                LHLD    VLOC            ;RESTORE OLD SP
8EF1    F9                      SPHL
8EF2    C3 AC 8B                JMP     IEND            ;ALL DONE
8EF5    80 00 00 01     FONE:   DB      200Q,0,0,001Q   ;FLOATING PNT ONE
                ;NEXT STATEMENT PROCESSOR
8EF9    2A 5B 04        NEXT:   LHLD    CPNT            ;FIX PNTR'S
8EFC    23                      INX     H
8EFD    23                      INX     H
8EFE    23                      INX     H
8EFF    0C                      INR     C
8F00    CD 00 86                CALL    ICP7
8F03    CD 25 82                CALL    ALPHA           ;LETTER?
8F06    D2 B5 8F                JNC     ER21            ;NO, ERROR
8F09    46                      MOV     B,M             ;YES, GET IT
8F0A    51                      MOV     D,C             ;SAVE C
8F0B    0E 00                   MVI     C,0             ;INIT C TO 0
8F0D    23                      INX     H               ;BUMP PNTR'S
```

```
8F0E  15                  DCR   D
8F0F  CA 1D 8F            JZ    NEXT1
8F12  CD 17 82            CALL  NUMB          ;NUMBER?
8F15  D2 B5 8F            JNC   ER21          ;NO, ERROR
8F18  4E                  MOV   C,M           ;YES, GET IT
8F19  15                  DCR   D             ;SHOULD BE EOL
8F1A  C2 B5 8F            JNZ   ER21
8F1D  21 00 00    NEXT1:  LXI   H,0           ;GET SP
8F20  39                  DAD   SP
8F21  22 8E 04            SHLD  VLOC          ;SAVE IT
8F24  2A 94 04            LHLD  NEST          ;GET NEST SP
8F27  7D                  MOV   A,L           ;COMPARE WITH BOTTOM
8F28  FE AA               CPI   BOTNS AND 377Q
8F2A  CA AB 8F            JZ    ER19          ;NEXT BEFORE FOR
8F2D  F9                  SPHL                ;LOAD SP
8F2E  E1                  POP   H             ;GET LAST INDEX
8F2F  78                  MOV   A,B           ;COMPARE TO CURRENT
8F30  BC                  CMP   H
8F31  C2 B0 8F            JNZ   ER20          ;NESTING ERROR
8F34  79                  MOV   A,C
8F35  BD                  CMP   L
8F36  C2 B0 8F            JNZ   ER20
8F39  2A 8E 04            LHLD  VLOC          ;ALL OK, RESTORE OLD SP
8F3C  F9                  SPHL
8F3D  3E 3F               MVI   A,77Q         ;MASK
8F3F  A0                  ANA   B             ;MASK OUT TOP 2 BITS
8F40  47                  MOV   B,A
8F41  CD 17 86            CALL  FSYM          ;FIND SYMBOL
8F44  EB                  XCHG                ;ADDRESS TO D,E
8F45  21 7C 04            LXI   H,FREG1       ;COPY STEP TO FREG1
8F48  CD C3 87            CALL  COPDH
8F4B  13                  INX   D             ;PNT TO CHARACTERISTIC OF STEP
8F4C  13                  INX   D
8F4D  13                  INX   D
8F4E  1A                  LDAX  D             ;GET IT
8F4F  E6 80               ANI   200Q          ;GET SIGN
8F51  17                  RAL                 ;ROTATE IT INTO CARRY
8F52  3F                  CMC                 ;COMPLEMENT IT
8F53  3E 00               MVI   A,0           ;MAKE SURE A=0
8F55  17                  RAL                 ;ROTATE TO LSB
8F56  3C                  INR   A             ;BUMP BY ONE
```

```
8F57  32 8E 04          STA     VLOC            ;SAVE IT, ITS =1 IF - STEP,
8F5A  13                INX     D               ;ELSE = 2  PNT TO VARIABLE PNTR
8F5B  EB                XCHG                    ;GET IT TO DE
8F5C  5E                MOV     E,M
8F5D  23                INX     H
8F5E  56                MOV     D,M
8F5F  23                INX     H
8F60  E5                PUSH    H               ;SAVE DATA BLOCK PNTR.
8F61  21 80 04          LXI     H,FREG2         ;COPY VARIBLE VALUE TO FREG2
8F64  CD C3 87          CALL    COPDH           ;SAVE VARIABLE LOCATION IN H,L
8F67  EB                XCHG
8F68  3E 02             MVI     A,2             ;SET UP TO ADD
8F6A  CD 5B 8A          CALL    BINOP           ;AND DO IT
8F6D  CD C3 87          CALL    COPDH           ;COPY TO VARIABLE
8F70  21 7C 04          LXI     H,FREG1         ;AND TO FREG1 FOR COMPARE
8F73  CD C3 87          CALL    COPDH
8F76  D1                POP     D               ;PNT TO LIMIT
8F77  21 80 04          LXI     H,FREG2         ;COPY TO FREG2
8F7A  CD C3 87          CALL    COPDH
8F7D  D5                PUSH    D               ;SAVE DATA BLOCK PNTR
8F7E  CD 43 8D          CALL    FCOMP           ;COMPARE
8F81  21 8E 04          LXI     H,VLOC          ;COMPARE WITH STEP TYPE
8F84  BE                CMP     M
8F85  E1                POP     H               ;GET DATA BLOCK PNTR.
8F86  CA 94 8F          JZ      NXTDN           ;YES => LOOP DONE
8F89  23                INX     H               ;LOOP NOT DONE
8F8A  23                INX     H               ;PNT TO TRANSFER ADD.
8F8B  23                INX     H
8F8C  23                INX     H
8F8D  5E                MOV     E,M             ;GET IT TO H,L
8F8E  23                INX     H
8F8F  56                MOV     D,M
8F90  EB                XCHG
8F91  C3 A4 86          JMP     ILOOP
8F94  21 94 04  NXTDN:  LXI     H,NEST          ;POP NEST STACK
8F97  34                INR     M
8F98  34                INR     M
8F99  C3 AC 8B          JMP     IEND            ;CONTINUE
8F9C  3E 16     ER16:   MVI     A,16H           ;'=' EXPECTED(NOTE: NO ARRAY
8F9E  C3 C8 85          JMP     ERROR           ;ELEMENTS FOR INDICES)
8FA1  3E 17     ER17:   MVI     A,17H           ;BAD SYNTAX NEAR 'TO' OR 'STEP'
```

```
8FA3    C3 C8 85                JMP     ERROR           ;IN FOR STATEMENT
8FA6    3E 18           ER18:   MVI     A,18H           ;FOR'S NESTED TOO DEEPLY
8FA8    C3 C8 85                JMP     ERROR
8FAB    3E 19           ER19:   MVI     A,19H           ;'NEXT' EXECUTED BEFORE A 'FOR'
8FAD    C3 C8 85                JMP     ERROR
8FB0    3E 20           ER20:   MVI     A,20H           ;NESTING ERROR, 'FOR'-'NEXT'
8FB2    C3 C8 85                JMP     ERROR
8FB5    3E 21           ER21:   MVI     A,21H           ;BAD INDEX IN FOR-NEXT
8FB7    C3 C8 85                JMP     ERROR
                        ;
                        ; THIS SUB CHECKS FOR PAGE BOUNDARY CROSSING
                        ; OF VARIABLE STORAGE BEFORE UPDATING
                        ; FORWARD POINTER
                        ;    D-E  POINT TO CURRENT LOCATION OF NEXT VARIABLE
                        ;    H-L  POINT TO PREVIOUS VARIABLE LOCATION
                        ;
                        ;  MODIFY D-E ( IF NECESSARY ) SO VARIABLE WILL NOT CROSS PAGE
                        ;  BOUNDARY
8FBA                    CHKLC:
8FBA    F5                      PUSH    PSW
8FBB    D5                      PUSH    D               ; SEE IF CURRENT VARIABLE
8FBC    3E 07                   MVI     A,7             ; STORAGE 8 WORD BLOCK
8FBE    83                      ADD     E               ; WILL CROSS PAGE BOUNDARY
8FBF    DA C5 8F                JC      CHOVL
                        ;  OK   -  DOES NOT CROSS PAGE
8FC2    D1                      POP     D
8FC3    F1                      POP     PSW
8FC4    C9                      RET
                        ;  PAGE BOUNDARY CROSSED  -  SET D-E TO START OF NEXT PAGE
8FC5                    CHOVL:
8FC5    D1                      POP     D
8FC6    14                      INR     D
8FC7    1E 00                   MVI     E,0
8FC9    F1                      POP     PSW
8FCA    C9                      RET
                        ;
                        ;  THIS SUB IS CALLED FROM 'DIM' PROCESSOR
                        ;  REGS. 'D-E' POINT TO NEXT AVAILABLE WORD OF VARIABLE STORAGE
                        ;  THIS SUB MAKES SURE THAT STORAGE STARTS ON A 4-WORD
                        ;  BOUNDARY SO FLT. PT. NUMBER WILL NOT CROSS PAGE
                        ;
```

```
8FCB                    CKDIM:
8FCB    7B                      MOV     A,E
8FCC    E6 03                   ANI     3
8FCE    C8                      RZ
8FCF    7B                      MOV     A,E
8FD0    E6 FC                   ANI     374Q
8FD2    C6 04                   ADI     4
8FD4    5F                      MOV     E,A
8FD5    7A                      MOV     A,D
8FD6    CE 00                   ACI     0
8FD8    57                      MOV     D,A
8FD9    C9                      RET

                        ; CALL ROUTINES

8FDA    AA 04           FWAM:   DW VEND                 ;DEFINE FWAM POINTER

                                END
NO PROGRAM ERRORS

                        SYMBOL TABLE

* 01

A       0007    ADDD    8A76    ADFLD   8B32    AFOND   892F
AGA     8A22    ALPHA   8225    AR      861E    ARCK    88D9
ARYES   88ED    ASBC    8A79    B       0000    BAC     8223
BINOP   8A5B    BMPTR   89DB    BND1    8576 *  BND2    85AD
BND3    85B0    BOTNS   04AA    BOUND   8557    BUPT    88B0 *
C       0001    C1      821C    CALLP   8D7A    CHOVL   8FC5
CHAR2   97F7    CHAR5   8517    CHEKE   89D1    CHK1    82A6
CHKLC   8FBA    CHKMD   89A7    CHPAR   0BB2    CKDIM   8FCB
CLSUB   8DEC    CONIN   0104    CONOU   0107    CONT    8796
CONV    97E5    COPD1   87C9    COPDH   87C3    CPNT    045B
CREG    0484    CVB     822D    CVB1    824E    CVB2    826B
D       0002    DCOMP   97F1    DFXL    97DC    DIM     8738
DIV     8AAD    DLOOP   873E    DOL     8A0C    E       0003
ECAV    89F1    ENDD    870A    ENTRY   867E    EOK     8A1A *
ER10    8C98    ER14    8CC3    ER16    8F9C    ER17    8FA1
ER18    8FA6    ER19    8FAB    ER20    8FB0    ER21    8FB5
ER6     8744    ER7     8C38    ER8     8A42    ER9     8888
```

```
ERLN    85E9    ERRET   8BE8    ERRØR   85C8    ERRR1   85DC
EVAL    89E7    EXPRE   8AF6    F1      845D    FALSE   8D32
FBAC    8683    FBILD   8E8B    FCØMP   8D43    FDAT    898F
FEXST   8EAD    FINPT   97E8    FIX     8935    FLDFD   8B4D
FLIMT   0490    FLØAT   97CA    FMULT   8AA7    FNDSB   8DAE
FØNE    8EF5    FØR     8E07    FØR10   8450    FØR11   844F
FØR12   844E    FØRM1   8457    FØRM2   8456    FØRM3   8455
FØRM4   8454    FØRM5   8458    FØRM6   845E    FØRM7   8453
FØRM8   8452    FØRM9   8451    FØUND   82A3    FPERR   8A7B
FREG1   047C    FREG2   0480    FSYM    8617    FWAM    8FDA
GET     87F4    GKEY    86DE    GØØD    8952    GØSUB   8DF0
GØTØ    8718    GREG    0477    GSENT   871B    GTRA    8721
H       0004    HLINP   0486    HØME    883F    HSRØR   0100
HSRIN   84E4    IAGA    8CAD    IBUF    0401    ICP2    860F
ICP4    860A    ICP7    8600    ICP8    89A3
IEND    8BAC    IFRT    8C9D    III     8CC8    ILØØP   86A4
INCPT   8611    INP     8993    INPER   8B70    INPØK   8BA9
INPUT   8B6C    INSER   8090    ISR10   816C    ISR11   8172
ISRI2   8178    ISR1A   80A3    ISRT1   80BE    ISRT2   80C5
ISRT3   80D5    ISRT4   80D8    ISRT5   8110    ISRT6   8122
ISRT7   8145    ISRT8   8149    ISRT9   8153    JTBL    86EE
KASE    0460    KDAT1   8358    KDAT2   836D    KDAT3   83A7
KDAT4   83C9    KDATA   8354    KFPNT   0456    KL2     0454
KL4     0456    KL6     0458    KLEN    0458    KLINE   0452
KØNT    8856    L       0005    L1      86C6    L2      828E
LADD    97DD    LDIV    97D6    LEN     0461    LENGT   8274
LET     8BFD    LIST    81EB    LIST    81D2    LMCM    97DF
LMUL    97D3    LØK     8C0D    LPNT    0452    LSUB    97D9
LUKØN   8631    M       0006    M1      8000    M1A     8013
M2      8017    M3      8020    M4      8043    M4A     8048
MCHK    97F4    MDBC    8AB0    MEMFU   851D    MEMST   0400
MESCR   0488    MINSE   8950    MØDE    0485    MØDE1   89B7
MØRE    889F    MULT    97EB    MULT1   0462    MULT2   0464
NDØU    8777    NEST    0494    NEXT    8EF9    NEXT1   8F1D
NFPER   8A9E    NL2     044D    NL4     044F    NL6     0451
NLE1    8278    NLE2    8287    NLINE   044B    NØENT   8650
NØLIN   8192    NØMAT   8645    NØRM    97C7    NØSYM   864E
NØT3    8D28    NSRCH   828B    NSTØK   8EE2    NUMB    8217
NUSUB   8D9D    NXTDN   8F94    NXTSP   0459    ØBUFF   0400
ØDA10   84DA    ØDAT1   8473    ØDAT2   8479    ØDAT3   8487
ØDAT4   8490    ØDAT5   849A    ØDAT6   84A0    ØDAT7   84AF
```

ØDAT8	84C6	ØDAT9	84D1	ØDATA	8469	ØKK	8862	
ØKLET	8749	ØKN	872D	ØNWD	8B45 *	ØUTR	87D6	*
P1	82BB	P2	82C5	PAD	82B0	PARLP	8DBD	
PEND	8B66	PIN1	84EA	PIN1A	84ED	PIN3	850F	
PINST	8C93	PK1	8427	PK2	8444	PL2	045B	*
PL4	045D	PL6	045F	PLINE	0459	PLØØP	8ACE	
PREXP	8DD5	PRI	8AB5	PRI1	8C83	PRMPT	8B7C	
PRTIT	8B0D	PSW	0006	PT1	806E	PTAPE	8066	
PTFIN	8C6C	PTVAL	97EE	QCHEK	8AE8	QØTØK	8ADD	
QSTR1	843F	QSTRG	843A	QTCHK	8211 *	QUITT	8209	
QUØTE	8AD4	RDKØN	886E	RELAT	8CF5	RETRN	8E05	
RINST	8851	RUN	8685	S2	8310 *	S3	8311	
S4	832C	S4A	832B	S5	8348	SAVV	8C52	
SBSAV	045D	SC1	8896	SCALR	88D1	SCØLN	8B27	
SCR	0466	SFSG	88B4	SINEQ	8D5B	SLØAD	88CA	
SØNWD	8B59	SØRCE	86B2	SP	0006	SPACE	89A1	
SPNT	0100	STAC	0496	STATU	010A	STLIN	0449	
STNER	8BEB	STØKK	8BCF	STØKV	8BC0	STP	8E76	
STPNT	818C	STRIN	8BB2	STSIZ	0014	STSPA	044B	
SUBAD	0102	SUBB	8AA1	SYMSR	8300	TAPE	81AB	
THEN	8D39	TIN1	83D5	TIN2	83FF	TIN4	8415	
TIN4A	841A	TIN5	840C	TIN5A	8411	TØPNS	0496	
TRUE	8D36	TTYIN	83D2	V1	8830	VALUE	87DE	
VAR	888D	VARAD	048A	VEND	04AA	VLØC	048E	
VNAME	048C	W1	82E2	W2	82F9	WFØR	8A47	
WFPER	8A97	WHAT	819F	WRIT	82D3	WRIT1	82D5	
ZRØL	97CD							

* 02

* 03

* 04

* 05

* 06

* 07

* 08

* 09

* 10

* 11

ODT

```
                                                ;ODT PROGRAM FOR THE MCS-80
                                                ;***********AUTHOR EUGENE R FISHER***********
                                                  ;4-20-74
0000                            ORG 0
0002                    TTY EQU 2
0006                    SND EQU 6
0003                    FLAG EQU 3
                        ;;
0000    31 FF 3F        ERROR:      LXI SP,37777Q       ;SET STACK POINTER,PAGE77,LOC 377
0003    3E BF                       MVI A,277Q          ;SEND A ?
0005    C3 3F 00                    JMP ERR
0008    C3 08 08                    JMP 4010Q            ;FIRST RESTART LOCATION
000B    00                          NOP
000C    00                          NOP
000D    00                          NOP
000E    00                          NOP
000F    00                          NOP
0010    C3 10 08                    JMP 4020Q
0013    00                          NOP
0014    00                          NOP
0015    00                          NOP
0016    00                          NOP
0017    00                          NOP
```

```
0018    C3 18 08                JMP 4030Q
001B    00                      NOP
001C    00                      NOP
001D    00                      NOP
001E    00                      NOP
001F    00                      NOP
0020    C3 20 08                JMP 4040Q
0023    00                      NOP
0024    00                      NOP
0025    00                      NOP
0026    00                      NOP
0027    00                      NOP
0028    C3 28 08                JMP 4050Q
002B    00                      NOP
002C    4D              PER:    MOV C,L             ;ROUTINE TO MAKE PERIOD CURRENT LOCATION
002D    C3 47 00                JMP NEXTC
0030    C3 E4 00        SEND:   JMP SEN             ;ENTRY POINT TO SEND ROUTINE
0033    CD F8 00        GO:     CALL CRLF           ;SEND A CRLF
0036    69                      MOV L,C             ;SET THE L REG
0037    E9                      PCHL        ;JMP TO STARTING ADDRESS
0038    E3              TRAP:   XTHL                ;CHANGE ORDER OF STACK H AND L LAST OUT
0039    D5                      PUSH D              ;FORTH OUT
003A    C5                      PUSH B               ;THIRD OUT
003B    F5                      PUSH    PSW             ;SECOND OUT
003C    E5                      PUSH H              ;FIRST OUT IS THE ADR WHENCE WE CAME
003D    3E D4                   MVI A,324Q  ;SEND A T
                                ;
                        ;THE STACK MAY BE DUMPED BY HITTING A CTRL-C TO GET THE
                        ; FOLLOWING SP+1  SP A F B C D E H L
                                ;
003F    F7              ERR:    RST SND
0040    CD F8 00                CALL CRLF
0043    AF              BEGIN:  XRA A               ;CLEAR THE AC
0044    16 04           BEG:    MVI D,4
0046    4F              SAV:    MOV C,A
0047    CD DB 00        NEXTC:  CALL READ
004A    78                      MOV A,B
004B    D6 B8                   SUI 270Q            ;IS IT A NUMBER
004D    F2 94 00                JP  TERM            ;270 MUST BE TERMINATOR OR ILLEGAL DIGIT
0050    78                      MOV A,B
0051    D6 B0                   SUI 260Q
```

```
0053    FA 94 00                JM   TERM          ;YES
0056    15                      DCR  D             ;>260 MUST BE DIGIT BUMP BUFFER CNT
0057    CA 00 00                JZ   ERROR         ;BUFFER OVERFLOW
005A    47                      MOV  B,A
005B    79                      MOV  A,C           ;GET PREVIOUS INPUTS
005C    17                      RAL
005D    17                      RAL
005E    17                      RAL
005F    DA 00 00                JC   ERROR         ;IF CARRY NUMBER WAS TOO BIG
0062    80                      ADD  B
0063    C3 46 00                JMP  SAV
0066    69              SLASH:  MOV  L,C           ;C CONTAINS ADDRESS
0067    7E              GETCON: MOV  A,M           ;GET CONTENTS
0068    CD C1 00                CALL OCTALP
006B    3E A0                   MVI  A,240Q
006D    F7                      RST  SND
006E    C3 43 00                JMP  BEGIN
0071    7A              LF:     MOV  A,D
0072    D6 04                   SUI  4
0074    CA 78 00                JZ   NINP          ;IF CNT STILL 4 NO INPUT WAS RECEIVED
0077    71                      MOV  M,C           ;IF CNT <4 DEPOSIT ONPUT IN MEMORY
0078    3E 8D           NINP:   MVI  A,215Q
007A    F7                      RST  SND
007B    2C                      INR  L             ;BUMP ADDRESS
007C    7C                      MOV  A,H           ;GET HI PART
007D    CD C1 00                CALL OCTALP        ;TYPE HI ADDRESS
0080    7D                      MOV  A,L           ;GET LOW PART
0081    CD C1 00                CALL OCTALP        ;TYPE LOW ORDER ADDRESS
0084    C3 67 00                JMP  GETCON
0087    CD F8 00        CR:     CALL CRLF          ;SEND A CRLF
008A    7A                      MOV  A,D
008B    D6 04                   SUI  4             ;BUFCNT =4 ?
008D    CA 43 00                JZ   BEGIN         ;YES NO INPUT SINCE LAST TERMINATOR
0090    71                      MOV  M,C           ;LOAD MEMORY WITH INPUT
0091    C3 43 00                JMP  BEGIN         ;ERF
0094    78              TERM:   MOV  A,B
0095    FE 8A                   CPI  212Q          ;IS IT A LF
0097    CA 71 00                JZ   LF
009A    FE 8D                   CPI  215Q          ;IS IT RETURN
009C    CA 87 00                JZ   CR
009F    FE D2                   CPI  322Q          ;R
```

```
00A1  CA 00 02                JZ 1000Q          ;START READER PROGRAM
00A4  FE AF                   CPI 257Q
00A6  CA 66 00                JZ  SLASH
00A9  FE C7                   CPI 307Q
00AB  CA 33 00                JZ  GO            ;G
00AE  FE AE                   CPI 256Q          ;PERIOD
00B0  CA 2C 00                JZ  PER
00B3  FE 83                   CPI 203Q
00B5  CA EF 00                JZ  CTRLC         ;CONTROL C
00B8  FE D3                   CPI 323Q          ;S
00BA  C2 00 00                JNZ ERROR         ;FOUND AN ERROR
00BD  61              SETX:   MOV H,C           ;SAME HI ADDRESS
00BE  C3 40 00                JMP ERR+1
00C1  06 04           OCTALP: MVI B,4           ;SET CNTR
00C3  07                      RLC
00C4  07                      RLC
00C5  5F                      MOV E,A           ;SAVE AC
00C6  3E A0                   MVI A,240Q        ;SEND A SPACE
00C8  F7                      RST SND
00C9  7B                      MOV A,E           ;GET THE AC
00CA  E6 03                   ANI 3             ;MASK
00CC  05              DECR:   DCR B             ;DECR
00CD  C8                      RZ                ;DONE ?
00CE  C6 B0                   ADI 260Q          ;NO
00D0  F7                      RST SND       ;SEND A CHARAACTER
00D1  7B                      MOV A,E           ;GET AC
00D2  07                      RLC
00D3  07                      RLC
00D4  07                      RLC
00D5  5F                      MOV E,A           ;SAVE AC
00D6  E6 07                   ANI 7             ;MASK
00D8  C3 CC 00                JMP DECR
00DB  DB 03           READ:   IN FLAG           ;ROUTINE TO READ ONE CHAR FROM TTY
00DD  1F                      RAR
00DE  D2 DB 00                JNC READ
00E1  DB 02                   IN TTY
00E3  47                      MOV B,A
00E4  D3 02           SEN:    OUT TTY           ;ROUTINE TO OUTPUT AN ASCII CHAR
00E6  DB 03           SEN1:   IN FLAG
00E8  1F                      RAR
00E9  1F                      RAR
```

```
00EA    78                      MOV A,B             ;RESTORE A REG FROM READ
00EB    D8                      RC                  ;DONE?
00EC    C3 E6 00                JMP SEN1            ;NO
00EF    E1              CTRLC:  POP H
00F0    7C                      MOV A,H
00F1    CD C1 00                CALL OCTALP
00F4    7D                      MOV A,L
00F5    C3 68 00                JMP GETCON+1
00F8    3E 8D           CRLF:   MVI A,215Q          ;SEND A CRLF
00FA    F7                      RST SND
00FB    3E 8A                   MVI A,212Q
00FD    C3 E4 00                JMP SEN             ;RETURN VIA SEND
                                END
NO PROGRAM ERRORS
```

SYMBOL TABLE

* 01

A	0007	B	0000	BEG	0044 *	BEGIN	0043
C	0001	CR	0087	CRLF	00F8	CTRLC	00EF
D	0002	DECR	00CC	E	0003	ERR	003F
ERROR	0000	FLAG	0003	GETCO	0067	GO	0033
H	0004	L	0005	LF	0071	M	0006
NEXTC	0047	NINP	0078	OCTAL	00C1	PER	002C
PSW	0006	READ	00DB	SAV	0046	SEN	00E4
SEN1	00E6	SEND	0030 *	SETX	00BD *	SLASH	0066
SND	0006	SP	0006	TERM	0094	TRAP	0038 *
TTY	0002						

```
                        ;       ////FLOATING POINT PACKAGE FOR THE MCS8
                        ;       ////BY DAVID MEAD
                        ;       ////MODIFIED BY HAL BRAND 9/6/74
                        ;       ////MODIFIED FOR 24 BIT MANTISSAS**********
                        ;       ////PLUS ADDED I/O CONVERSION ROUTINES
                        ;       ////NEW ROUTINE COMMENTS
                        ;       ////ARE PRECEEDED BY /
                        ;       ////OTHER CHANGES ARE NOTED BY **
                        ;       ////MODIFIED BY FRANK OLKEN 6/26/75
                        ;
                        ;
9000                            ORG 110000Q
                        ;
                        ;
0104                    CONIN   EQU     404Q        ; JMP TABLE LOCATION OF CONSOLE INP.
010A                    STATUS  EQU     412Q        ; JMP TABLE LOC. FOR STATUS PORT INPUT
97FD                    OUTR    EQU     113775Q             ;LINK TO BASIC
87D6                    OUTL    EQU     103726Q
8993                    INL     EQU     104623Q
97FA                    INP     EQU     113772Q             ;LINK TO BASIC
00C0                    MINCH   EQU     300Q        ;MINIMUM CHARACTERISTIC WITH SIGN EXTENDED
003F                    MAXCH   EQU     077Q        ;MAXIMUM CHARACTERISTIC WITH SIGN EXTENDED
                        ;
                        ;*****************************************************
                        ;       //// DIVIDE SUBROUTINE
                        ;*****************************************************
                        ;
9000    CD 69 93        LDIV:   CALL    CSIGN       ;COMPUTE SIGN OF RESULT
9003    CD DA 91                CALL    ZCHK        ;CHECK IF DIVIDEND = ZERO
9006    C2 12 90                JNZ     DTST2       ;IF DIVIDEND .NE. 0 CHECK DIVISOR
9009    CD E2 91                CALL    BCHK        ;CHECK FOR ZERO/ZERO
900C    CA A8 92                JZ      INDFC       ;ZERO/ZERO = INDEFINITE
900F    C3 AF 92                JMP     WZERC       ;ZERO/NONZERO = ZERO
9012    CD E2 91        DTST2:  CALL    BCHK        ;COME HERE IF DIVIDEND .NE. 0
9015    CA 5B 93                JZ      OFLWC       ;NONZERO/ZERO = OVERFLOW
                                                    ;IF WE GET HERE, THINGS LOOK OKAY
```

```
9018  5D                     MOV     E,L       ;SAVE BASE IN E
9019  69                     MOV L,C       ;BASE 6 TO L
901A  CD 1D 92               CALL DCLR     ;CLEAR QUOTIENT MANTISSA SLOT
901D  6B                     MOV L,E       ;RESTORE BASE IN L
901E  CD 10 93               CALL ENT1     ;DO FIRST CYCLE
9021  69                     MOV L,C       ;BASE  6 TO L
9022  CD E9 91               CALL DLST     ;MOVE QUOTIENT OVER ONE PLACE
9025  16 17                  MVI D,23      ;NUMBER OF ITERATIONS TO D
9027  6B            REP3:    MOV L,E
9028  CD 0A 93               CALL ENT2
902B  15                     DCR D         ;DEC D
902C  CA 3B 90               JZ  GOON
902F  7D                     MOV A,L
9030  69                     MOV L,C       ;BASE 6 TO L
9031  4F                     MOV C,A
9032  CD E9 91               CALL DLST     ;MOVE QUOTIENT MANT OVER
9035  7D                     MOV A,L       ;CPTR TO A
9036  59                     MOV E,C       ;LPTR TO E
9037  4F                     MOV C,A       ;CPTR TO C
9038  C3 27 90               JMP REP3
                ;
903B  CD E1 92       GOON:    CALL    AORS          ;CHECK IF RESULT IS NORMALIZED
903E  FA 4D 90               JM  CRIN
9041  7D                     MOV A,L       ;LPTR TO A
9042  69                     MOV L,C       ;CPTR TO L
9043  4F                     MOV C,A       ;LPTR TO C
9044  CD E9 91               CALL DLST     ;SHIFT QUOTIENT LEFT
9047  4D                     MOV C,L
9048  6B                     MOV L,E
9049  CD 39 93               CALL    LDCP    ;COMPUTE THE CHARACTERISTIC OF RESULT
904C  C9                     RET
                ;
904D  CD 4C 92       CRIN:    CALL    CFCHE         ;GET A=CHAR(H,L), E=CHAR(H,B)
9050  93                     SUB     E       ;NEW CHAR = CHAR(DIVIDEND) -
9051  FE 7F                  CPI     177Q    ;CHAR(DVISIOR) CHECK MAX POSITIVE NUMBER
9053  CA 5B 93               JZ      OFLWC   ;JUMP ON OVERFLOW
9056  C6 01                  ADI     1       ;ADD 1 SINCE WE DID NOT LEFTSHIFT
9058  CD 44 93               CALL    CCHK    ;CHECK AND STORE CHARACTERISTIC
905B  C9                     RET             ;RETURN
                ;
                ;
```

```
                        ;
                        ;*******************************************************
                        ;       //// ADDITION SUBROUTINE
                        ;*******************************************************
                        ;
905C    AF              LADD:   XRA A               ;/***SET UP TO ADD
905D    C3 62 90                JMP LADS            ;/NOW DO IT
                        ;
                        ;*******************************************************
                        ;       //// SUBTRACTION SUBROUTINE
                        ;*******************************************************
                        ;
9060    3E 80           LSUB:   MVI A,200Q            ;/****SET UP TO SUBTRACT
                                                    SUBROUTINE LADS
                                                    FLOATING POINT ADD OR SUB
                                                    A 128 ON ENTRY SUB
                                                    A 0 ON ENTRY ADD
                                                    F-S F,FIRST OPER DESTROYED
                                                    BASE  11 USED FOR SCRATCH
9062    CD EF 92        LADS:   CALL ACPR             ;SAVE ENTRY PNT AT BASE  6
9065    CD E2 91                CALL    BCHK            ;CHECK ADDEND/SUBTRAHEND = ZERO
9068    C8                      RZ                      ;IF SO, RESULT=ARG SO RETURN
                                                        ;THIS WILL PREVENT UNDERFLOW
                                                        ;INDICATION ON ZERO + OR - ZERO
9069    CD 5B 92                CALL CCMP
906C    CA 9C 90                JZ  EQ02            ;IF EQUAL, GO ON
906F    57                      MOV D,A             ;SAVE LPTR CHAR IN D
9070    DA 7F 90                JC  LLTB
9073    93                      SUB E               ;L.GT.B IF HERE
9074    E6 7F                   ANI 127
9076    57                      MOV D,A             ;DIFFERENCE TO D
9077    5D                      MOV E,L             ;SAVE BASE IN E
9078    69                      MOV L,C             ;C PTR TO L
9079    2C                      INR L               ;C PTR 1 TO L
907A    73                      MOV M,E             ;SAVE BASE IN C PTR 1
907B    68                      MOV L,B             ;B PTR TO L
907C    C3 84 90                JMP NCHK
907F    7B              LLTB:   MOV A,E             ;L.LT.B IF HERE,BPTR TO A
```

```
9080    92                      SUB D           ;SUBTRACT LPTR CHAR FROM BPTR CHAR
9081    E6 7F                   ANI 127
9083    57                      MOV D,A         ;DIFFERENCE TO D
9084    3E 18           NCHK:   MVI A,24
9086    BA                      CMP D
9087    D2 8C 90                JNC SH10
908A    16 18                   MVI D,24
908C    B7              SH10:   ORA A
908D    CD F8 91                CALL DRST
9090    15                      DCR D
9091    C2 8C 90                JNZ SH10
9094    7D              EQUL:   MOV A,L
9095    B8                      CMP B
9096    C2 9C 90                JNZ EQ02        ;F.GT.S IF L.NE.B
9099    69                      MOV L,C         ;C PTR TO L
909A    2C                      INR L           ;C PTR 1 TO L
909B    6E                      MOV L,M         ;RESTORE L
909C    CD 02 91        EQ02:   CALL LASD       ;CHECK WHAT TO
909F    CD EF 92                CALL ACPR       ;SAVE ANSWER
90A2    FE 02                   CPI 2           ;TEST FOR ZERO ANSWER
90A4    C2 AA 90                JNZ NOTO
90A7    C3 8D 92                JMP     WZER      ;WRITE FLOATING ZERO AND RETURN
                ;
90AA    16 01           NOTO:   MVI D,1         ;WILL TEST FOR SUB
90AC    A2                      ANA D
90AD    CA D6 90                JZ  ADDZ        ;LSB 1 INPLIES SUB
90B0    CD E7 92                CALL TSTR       ;CHECK NORMAL/REVERSE
90B3    CA B9 90                JZ  SUBZ        ;IF NORMAL,GO SUBZ
90B6    7D                      MOV A,L         ;OTHERWISE REVERSE
90B7    68                      MOV L,B         ;ROLES
90B8    47                      MOV B,A         ;OF L AND B
                ;
90B9    CD 26 92        SUBZ:   CALL    DSUB      ;SUBTRACT SMALLER FROM BIGGER
90BC    CD EF 90                CALL    MANT      ;SET UP SIGN OF RESULT
90BF    CD E7 92                CALL    TSTR      ;SEE IF WE NEED TO INTERCHANGE
                                                  ;BPTR AND LPTR
90C2    CA AD 91                JZ      NORM      ;NO INTERCHANGE NECESSARY, SO NORMALIZE
                                                  ;AND RETURN
90C5    7D                      MOV A,L         ;INTERCHANGE
90C6    68                      MOV L,B         ;L
90C7    47                      MOV B,A         ;AND B
```

```
90C8  79                  MOV A,C           ;CPTR  TO A
90C9  48                  MOV C,B           ;BPTR TO C
90CA  5D                  MOV E,L           ;LPTR TO E
90CB  47                  MOV B,A           ;CPTR TO B
90CC  CD 24 93            CALL LXFR         ;MOVE BPTR> TO  LPTR>
90CF  7?                  MOV A,B
90D0  41                  MOV B,C
90D1  4F                  MOV C,A
90D2  6B                  MOV L,E
90D3  C3 AD 91            JMP     NORM      ;NORMALIZE RESULT AND RETURN
                    ;
                    ;   COPY THE LARGER CHARACTERISTIC TO THE RESULT
                    ;
90D6  CD 5B 92      ADD2:     CALL    CCMP      ;COMPARE THE CHARACTERISTICS
90D9  D2 DF 90                JNC     ADD2      ;IF CHAR(H,L) .GE. CHAR(H,B) CONTINUE
90DC  CD 8D 93                CALL    BCTL      ;IF CHAR(H,L) .LT. CHAR(H,B) THE COPY
                                                ;CHAR(H,B) TO CHAR(H,L)
90DF  CD EF 90      ADD2:     CALL    MANT      ;COMPUTE SIGN OF RESULT
90E2  CD 06 92                CALL    DADD      ;ADD MANTISSAS
90E5  D2 D2 92                 JNC    SCCFG          ;IF THERE IS NO OVFLW - DONE
90E8  CD F8 91                CALL    DRST      ;IF OVERFLOW SHIFT RIGHT
90EB  CD B6 92                CALL    INCR      ;AND INCREMENT CHARACTERISTIC
90EE  C9                      RET               ;ALL DONE, SO RETURN
                    ;
                    ;   THIS ROUTINE STORES THE MANTISSA SIGN IN THE RESULT
                    ;   THE SIGN HAS PREVIOUSLY BEEN COMPUTED BY LASD.
                    ;
90EF  5D            MANT:     MOV E,L           ;SAVE L PTR
90F0  69                      MOV L,C           ;C PTR TO L
90F1  7E                      MOV A,M           ;LOAD INDEX WORD
90F2  E6 80                   ANI 128           ;SCARF SIGN
90F4  6B                      MOV L,E           ;RESTORE L PTR
90F5  2C                      INR L             ;L PTR 2
90F6  2C                      INR L
90F7  2C                      INR L             ;TO L
90F8  5F                      MOV E,A           ;SAVE SIGN IN E
90F9  7E                      MOV A,M
90FA  E6 7F                   ANI 127           ;SCARF CHAR
90FC  83                      ADD E             ;ADD SIGN
90FD  77                      MOV M,A           ;STORE IT
90FE  2D                      DCR L             ;RESTORE
```

```
90FF  2D          DCR L
9100  2D          DCR L          ;L PTR
9101  C9          RET
                ;
                ;
                ;                SUBROUTINE LASD
                ;                UTILITY ROUTINE FOR LADS
                ;                CALCULATES TRUE OPER AND SGN
                ;                RETURNS ANSWER IN
9102  CD 79 93  LASD:  CALL MSFH    ;FETCH MANT SIGNS, F IN A,D
9105  BB          CMP E            ;COMPARE SIGNS
9106  DA 34 91    JC  ABCH         ;F ,S- MEANS GO TO A BRANCH
9109  C2 3D 91    JNZ BBCH         ;F- S  MEANS GO TO B BRANCH
910C  83          ADD E            ;SAME SIGN IF HERE, ADD SIGNS
910D  DA 22 91    JC  BMIN         ;IF BOTH MINUS, WILL OVERFLOW
9110  CD E1 92    CALL AORS        ;BOTH POS IF HERE
9113  F2 46 91    JP  L000         ;IF AN ADD, LOAD 0
9116  CD F4 92    COM1: CALL DCMP  ;COMPARE F WITH S
9119  DA 54 91    JC  L131         ;S.GT.F,SO LOAD 131
911C  C2 48 91    JNZ L001         ;F.GT.S,SO LOAD 1
911F  3E 02       L002: MVI A,2    ;ERROR CONDITION, ZERO ANSWER
9121  C9          RET
9122  CD E1 92    BMIN: CALL AORS  ;CHECK FOR ADD OR SUB
9125  F2 4E 91    JP  L128         ;ADD, SO LOAD 128
9128  CD F4 92    COM2: CALL DCMP  ;COMPARE F WITH S
912B  DA 4B 91    JC  L003         ;S.GT.F,SO LOAD 3
912E  C2 51 91    JNZ L129         ;FGT.S.SO LOAD 129
9131  C3 1F 91    JMP L002         ;ERROR
9134  CD E1 92    ABCH: CALL AORS  ;FT,S- SO TEST FOR A/S
9137  FA 46 91    JM  L000         ;SUBTRACT, SO LOAD 0
913A  C3 16 91    JMP COM1         ;ADD, SO GO TO DCMP
913D  CD E1 92    BBCH: CALL AORS  ;F-,S ,SO TEST FOR A/S
9140  FA 4E 91    JM  L128         ;SUB
9143  C3 28 91    JMP COM2         ;ADD
9146  AF          L000: XRA A
9147  C9          RET
9148  3E 01       L001: MVI A,1
914A  C9          RET
914B  3E 03       L003: MVI A,3
914D  C9          RET
914E  3E 80       L128: MVI A,128
```

```
9150    C9                      RET
9151    3E 81           L129:   MVI A,129
9153    C9                      RET
9154    3E 83           L131:   MVI A,131
9156    C9                      RET
                        ;
                        ;                       SUBROUTINE LMCM
                        ;                       COMPARES THE MAGNITUDE OF
                        ;                       TWO FLOATING PNT NUMBERS
                        ;                       Z 1 IF   ,C 1 IF F.LT.S.
9157    CD 5B 92        LMCM:   CALL CCMP         ;CHECK CHARS
915A    C0                      RNZ               ;RETURN IF NOT EQUAL
915B    CD F4 92                CALL DCMP         ;IF EQUAL, CHECK MANTS
915E    C9                      RET
                        ;
                        ;
                        ;*************************************************
                        ;       //// MULTIPLY SUBROUTINE
                        ;*************************************************
                        ;
                        ;                       SUBROUTINE LMUL
                        ;                       FLOATING POINT MULTIPLY
                        ;                       L PTR X B PTR TO C PTR
                        ;
915F    CD 69 93        LMUL:   CALL    CSIGN       ;COMPUTE SIGN OF RESULT AND STORE IT
9162    CD DA 91                CALL    ZCHK        ;CHECK FIRST OPERAND FOR ZERO
9165    CA AF 92                JZ      WZERC       ;ZERO * ANYTHING = ZERO
9168    CD E2 91                CALL    BCHK        ;CHECK SECOND OPERAND FOR ZERO
916B    CA AF 92                JZ      WZERC       ;ANYTHING * ZERO = ZERO
916E    5D                      MOV E,L           ;SAVE L PTR
916F    69                      MOV L,C           ;C PTR TO L
9170    CD 1D 92                CALL DCLR         ;CLR PRODUCT MANT LOCS
9173    6B                      MOV L,E           ;L PTR TO L
9174    16 18                   MVI D,24          ;LOAD NUMBER ITERATIONS
9176    CD F8 91        KPGO:   CALL DRST         ;SHIFT L PTR RIGHT
9179    DA A4 91                JC  MADD          ;WILL ADD B PTR IF C 1
917C    7D                      MOV A,L           ;INTERCHANGE
917D    69                      MOV L,C           ;L AND
917E    4F                      MOV C,A           ;C PTRS
917F    CD F8 91        INTR:   CALL DRST         ;SHIFT PRODUCT OVER
```

```
9182    7D                      MOV A,L             ;INTERCHANGE
9183    69                      MOV L,C             ;L AND C PTRS BACK TO
9184    4F                      MOV C,A             ;ORIGINAL>
9185    15                      DCR D
9186    C2 76 91                JNZ KPGO            ;MORE CYCLES IF Z 0
9189    CD E1 92                CALL    AORS          ;TEST IF RESULT IS NORMALIZED
918C    FA 40 93                JM      LMCP          ;IF NORMALIZED GO COMPUTE CHAR
918F    5D                      MOV     E,L           ;SAVE LPTR IN E
9190    69                      MOV     L,C           ;SET L=CPTR
9191    CD E9 91                CALL    DLST          ;LEFT SHIFT RESULT TO NORMALIZE
9194    6B                      MOV     L,E           ;RESTORE LPTR
9195    CD 4C 92                CALL    CFCHE         ;OTHERWISE SET A=CHAR(H,L), E=CHAR(H,B)
9198    83                      ADD     E             ;CHAR(RESULT) = CHAR(H,L) + CHAR(H,B)
9199    FE 80                   CPI     200Q          ;CHECK FOR SMALLEST NEGATIVE NUMBER
919B    CA 62 93                JZ      UFLWC         ;IF SO THEN UNDERFLOW
919E    D6 01                   SUI     1             ;SUBTRACT 1 TO COMPENSATE FOR NORMALIZE
91A0    CD 44 93                CALL    CCHK          ;CHECK CHARACTERISTIC AND STORE IT
91A3    C9                      RET                   ;RETURN
                ;
91A4    7D              MADD:   MOV A,L             ;INTERCHANGE
91A5    69                      MOV L,C             ;L AND
91A6    4F                      MOV C,A             ;C PTRS
91A7    CD 06 92                CALL DADD           ;ACCUMULATE PRODUCT
91AA    C3 7F 91                JMP INTR
                ;
                ;   SUBROUTINE NORM
                ;
                ;       THIS SUBROUTINE WILL NORMALIZE A FLOATING POINT
                ;       NUMBER, PRESERVING ITS ORIGINAL SIGN.
                ;       WE CHECK FOR UNDERFLOW AND SET THE CONDITION
                ;       FLAG APPROPRIATELY.  (SEE ERROR RETURNS).
                ;       THER IS AN ENTRY POINT TO FLOAT A SIGNED INTEGER
                ;       (FLOAT) AND AN ENTRY POINT TO FLOAT AN UNSIGNED
                ;       INTEGER.
                ;
                ;   ENTRY POINTS:
                ;
                ;       NORM  - NORMALIZE FLOATING PT NUMBER AT (H,L)
                ;       FLOAT - FLOAT TRIPLE PRECISION INTEGER AT (H,L)
                ;               PRESERVING SIGN BIT IN (H,L)+3
                ;       DFXL  - FLOAT UNSIGNED (POSITIVE) TRIPLE PRECISION
```

```
                        ;                AT (H,L)
                        ;
                        ;REGISTERS ON EXIT:
                        ;
                        ;       A = CONDITION FLAG (SEE ERROR RETURNS)
                        ;       D,E = GARBAGE
                        ;       B,C,H,L = SAME AS ON ENTRY
                        ;
91AD    5D              NORM:   MOV     E,L             ;SAVE L IN E
91AE    CD 41 92        NORM1:  CALL    GCHAR           ;GET CHAR(H,L) IN A WITH SIGN EXTENDED
91B1    57                      MOV     D,A             ;SAVE CHAR IN D
91B2    6B              FXL1:   MOV     L,E             ;RESTORE L
91B3    CD DA 91        FXL2:   CALL    ZMCHK           ;CHECK FOR ZERO MANTISSA
91B6    CA 8D 92                JZ      WZER            ;IF ZERO MANTISSA THEN ZERO RESULT
91B9    7E              REP6:   MOV     A,M             ;GET MOST SIGNIFICANT BYTE OF
                                                        ;MANTISSA
91BA    B7                      ORA     A               ;SET FLAGS
91BB    FA CB 91                JM      SCHAR           ;IF MOST SIGNFICANT BIT = 1 THEN
                                                        ;NUMBER IS NORMALIZED AND WE GO TO
                                                        ;STORE THE CHARACTERISTIC
91BE    7A                      MOV     A,D             ;OTHERWISE CHECK FOR UNDERFLOW
91BF    FE C0                   CPI     MINCH           ;COMPARE WITH MINIMUM CHAR
91C1    CA 63 92                JZ      WUND            ;IF EQUAL THEN UNDERFLOW
91C4    CD E9 91                CALL    DLST            ;SHIFT MANTISSA LEFT
91C7    15                      DCR     D               ;DECREMENT CHARACTERSTIC
91C8    C3 B9 91                JMP     REP6            ;LOOP AN TEST NEXT BIT
91CB    C3 C3 92        SCHAR:  JMP     INCR3           ;STORE THE CHARACTERISTIC USING
                                                        ;THE SAME CODE AS THE INCREMENT
                        ;
91CE    5D              DFXL:   MOV     E,L             ;ENTER HERE TO FLOAT UNSIGNED
                                                        ;INTEGER
                                                        ;FIRT SAVE L IN E
91CF    2C                      INR     L               ;MAKE (H,L) POINT TO CHAR
91D0    2C                      INR     L               ;MAKE (H,L) POINT TO CHAR
91D1    2C                      INR     L               ;MAKE (H,L) POINT TO CHAR
91D2    AF                      XRA     A               ;ZERO ACCUMULATOR
91D3    77                      MOV     M,A             ;STORE A PLUS (+) SIGN
91D4    6B                      MOV     L,E             ;RESTORE L
91D5    16 18           FLOAT:  MVI     D,24            ;ENTER HERE TO FLOAT INTEGER
                                                        ;PRESERVING ORIGINAL SIGN IN (H,L)+3
                                                        ;SET UP CHARACTERISTIC
91D7    C3 B3 91                JMP     FXL2            ;GO FLOAT THE NUMBER
```

```
                        ;
                        ;
                        ;   SUBROUTINE ZCHK
                        ;
                        ;       THIS ROUTINE SETS THE ZERO FLAG IF IT DETECTS
                        ;       A FLOATING ZERO AT (H,L).
                        ;
                        ;   SUBROUTINE ZMCHK
                        ;
                        ;       THIS ROUTINE SETS THE ZERO FLAG IF IT DETECTS A
                        ;       ZERO MANTISSA AT (H,L)
                        ;
91DA                    ZCHK:
91DA    2C              ZMCHK:  INR     L               ;SET L TO POINT LAST BYTE OF MANTISSA
91DB    2C                      INR     L               ;SET L TO POINT TO LAST BYTE OF MANTISSA
91DC    7E                      MOV     A,M             ;LOAD LEAST SIGNIFICANT BYTE
91DD    2D                      DCR     L               ;L POINTS TO MIDDLE BYTE
91DE    B6                      ORA     M               ;OR WITH LEAST SIGNFICANT BYTE
91DF    2D                      DCR     L               ;L POINTS TO MOST SIGNFICANT BYTE
                                                        ;OF MANTISSA (ORIGINAL VALUE)
91E0    B6                      ORA     M               ;OR IN MOST SIGNFICANT BYTE
91E1    C9                      RET                     ;RETURNS WITH ZERO FLAG SET APPROPRIATELY
                        ;
                        ;   SUBROUTINE BCHK
                        ;
                        ;       THIS ROUTINE CHECKS (H,B) FOR FLOATING PT ZERO
                        ;
91E2    5D              BCHK:   MOV     E,L             ;SAVE LPTR IN E
91E3    68                      MOV     L,B             ;SET L=BPTR
91E4    CD DA 91                CALL    ZCHK            ;CHECK FOR ZERO
91E7    6B                      MOV     L,E             ;RESTORE L=LPTR
91E8    C9                      RET                     ;RETURN
                        ;
                        ;                               SUBROUTINE DLST
                        ;                               SHIFTS DBL WORD ONE PLACE LF
91E9    2C               DLST:  INR L
91EA    2C                      INR L                     ;/***TP
91EB    7E                      MOV A,M                   ;LOAD IT
```

```
91EC  B7          ORA A          ;KILL CARRY
91ED  17          RAL            ;SHIFT IT LEFT
91EE  77          MOV M,A        ;STORE IT
91EF  2D          DCR L
91F0  7E          MOV A,M        ;LOAD IT
91F1  17          RAL            ;SHIFT IT LEFT
                  ;              IF CARRY SET BY FIRST SHIFT
                  ;              IT WILL BE IN LSB OF SECOND
91F2  77          MOV M,A
91F3  2D          DCR L          ;/***TP EXTENSION
91F4  7E          MOV A,M
91F5  17          RAL
91F6  77          MOV M,A        ;/***ALL DONE TP
91F7  C9          RET
                  ;
                  ;              SUBROUTINE DRST
                  ;              SHIFTS DOUBLE WORD ONE PLACE
                  ;              TO THE RIGHT
                  ;              DOES NOT AFFECT D
91F8  5D  DRST:   MOV E,L        ;/***TP MODIFIED RIGHT SHIFT TP
91F9  7E          MOV A,M        ;LOAD FIRST WORD
91FA  1F          RAR            ;ROTATE IT RIGHT
91FB  77          MOV M,A        ;STORE IT
91FC  2C          INR L          ;/*** TP
91FD  7E          MOV A,M        ;LOAD SECOND WORD
91FE  1F          RAR            ;SHIFT IT RIGHT
91FF  77          MOV M,A        ;STORE IT
9200  2C          INR L          ;/*** TP EXTENSION
9201  7E          MOV A,M
9202  1F          RAR
9203  77          MOV M,A
9204  6B          MOV L,E
9205  C9          RET            ;/***TP - ALL DONE TP
                  ;
                  ;              SUBROUTINE DADD
                  ;              ADDS TWO DOUBLE PRECISION
                  ;              WORDS, C 1 IF THERE IS OVRFLW
9206  5D  DADD:   MOV E,L        ;SAVE BASE IN E
9207  68          MOV L,B        ;BASE  3 TO L
9208  2C          INR L          ;BASE  4 TO L
9209  2C          INR L          ;/***TP
920A  7E          MOV A,M        ;LOAD S MANTB
920B  6B          MOV L,E        ;BASE TO L
```

```
920C  2C          INR L          ;BASE  1 TO L
920D  2C          INR L          ;/***TP
920E  86          ADD M          ;ADD TWO MANTB S
920F  77          MOV M,A        ;STORE ANSWER
9210  68          MOV L,B        ;/***TP EXTENSION
9211  2C          INR L
9212  7E          MOV A,M
9213  6B          MOV L,E
9214  2C          INR L
9215  8E          ADC M
9216  77          MOV M,A        ;/***TP - ALL DONE
9217  68          MOV L,B        ;BASE  3 TO L
9218  7E          MOV A,M        ;MANTA OF S TO A
9219  6B          MOV L,E        ;BASE TO L
921A  8E          ADC M          ;ADD WITH CARRY
921B  77          MOV M,A        ;STORE ANSWER
921C  C9          RET
                  ;
                  ;              SUBROUTINE DCLR
                  ;              CLEARS TWO SUCCESSIVE
                  ;              LOCATIONS OF MEMORY
921D  AF    DCLR: XRA A
921E  77          MOV M,A
921F  2C          INR L
9220  77          MOV M,A
9221  2C          INR L          ;/***TP EXTENSION
9222  77          MOV M,A        ;/***TP ZERO 3
9223  2D          DCR L          ;/***TP - ALL DONE
9224  2D          DCR L
9225  C9          RET
                  ;              /*****ALL NEW DSUB - SHORTER***
                  ;              SUBROUTINE DSUB
                  ;              DOUBLE PRECISION SUBTRACT
9226  5D    DSUB: MOV E,L        ;SAVE BASE IN E
9227  2C          INR L          ;/***TP EXTENSION
9228  2C          INR L          ;/START WITH LOWS
9229  7E          MOV A,M        ;/GET ARG
922A  68          MOV L,B        ;/NOW SET UP TO SUB
922B  2C          INR L
922C  2C          INR L
922D  96          SUB M          ;/NOW DO IT
922E  6B          MOV L,E        ;/NOW MUST PUT IT BACK
```

```
922F  2C          INR L
9230  2C          INR L
9231  77          MOV M,A          ;/PUT BACK
9232  2D          DCR L            ;/***TP - ALL DONE
9233  7E          MOV A,M          ;/GET LOW OF LOP
9234  68          MOV L,B          ;/SET TO BOP
9235  2C          INR L            ;/SET TO BOP LOW
9236  9E          SBB M            ;/GET DIFF. OF LOWS
9237  6B          MOV L,E          ;/SAVE IN LOP LOW
9238  2C          INR L            ;/TO LOP LOW
9239  77          MOV M,A          ;/INTO RAM
923A  2D          DCR L            ;/BACK UP TO LOP HIGH
923B  7E          MOV A,M          ;/GET LOP HIGH
923C  68          MOV L,B          ;/SET TO BOP HIGH
923D  9E          SBB M            ;/SUB. WITH CARRY
923E  6B          MOV L,E          ;/SAVE IN LOP HIGH
923F  77          MOV M,A          ;/INTO RAM
9240  C9          RET              ;/ALL DONE - MUCH SHORTER

                  ;
                  ;   SUBROUTINE GCHAR
                  ;
                  ;       THIS SUBROUTINE RETURNS THE CHARACTERISTIC OF
                  ;       THE FLOATING POINT NUMBER POINTED TO BY (H,L)
                  ;       IN THE A REGISTER WITH ITS SIGN EXTENDED INTO THE
                  ;       LEFTMOST BIT.
                  ;
                  ;   REGISTERS ON EXIT:
                  ;
                  ;       A = CHARACTERISTIC OF (H,L) WITH SIGN EXTENDED
                  ;       L = (ORIGINAL L) + 3
                  ;       B,C,D,E,H = SAME AS ON ENTRY
                  ;
9241  2C          GCHAR:  INR     L        ;MAKE (H,L) POINT TO CHAR
9242  2C                  INR     L        ;MAKE (H,L) POINT TO CHAR
9243  2C                  INR     L        ;MAKE (H,L) POINT TO CHAR
9244  7E                  MOV     A,M      ;SET A=CHAR + MANTISSA SIGN
9245  E6 7F               ANI     177Q     ;GET RID OF MANTISSA SIGN BIT
9247  C6 40               ADI     100Q     ;PROPAGATE CHAR SIGN INTO LEFTMOST BIT
9249  EE 40               XRI     100Q     ;RESTORE ORIGINAL CHAR SIGN BIT
924B  C9                  RET              ;RETURN WITH (H,L) POINTING TO THE
                                           ;CHAR = ORIGINAL (H,L)+3
                                           ;SOMEONE ELSE WILL CLEAN UP
```

```
;
;   SUBROUTINE CFCHE
;
;       THIS SUBROUTINE RETURNS THE CHARACTERISTICS OF THE
;       FLOATING POINT NUMBERS POINTED TO BY (H,L) AND
;       (H,B) IN THE A AND E REGISTERS RESPECTIVELY,
;       WITH THEIR SIGNS EXTENDED INTO THE LEFTMOST BIT.
;
;   REGISTERS ON EXIT:
;
;       A = CHARACTERISTIC OF (H,L) WITH SIGN EXTENDED
;       E = CHARACTERISTIC OF (H,B) WITH SIGN EXTENDED
;       B,C,H,L = SAME AS ON ENTRY
;       D = A
;
924C  5D          CFCHE:  MOV     E,L             ;SAVE LPTR IN E
924D  68                  MOV     L,B             ;SET L = BPTR
924E  CD 41 92            CALL    GCHAR           ;GET CHAR(H,B) WITH SIGN EXTENDED IN A
9251  6B                  MOV     L,E             ;RESTORE L = LPTR
9252  5F                  MOV     E,A             ;SET E=CHAR(H,B) WITH SIGN EXTENDED
9253  CD 41 92            CALL    GCHAR           ;SET A=CHAR(H,L) WITH SIGN EXTENDED
9256  2D                  DCR     L               ;RESTORE L = LPTR
9257  2D                  DCR     L               ;RESTORE L = LPTR
9258  2D                  DCR     L               ;RESTORE L = LPTR
9259  57                  MOV     D,A             ;SET D=A=CHAR(H,L) WITH SIGN EXTENDED
925A  C9                  RET
;
;   SUBROUTINE CCMP
;
;       THIS SUBROUTINE COMPARES THE CHARACTERISTICS OF
;       FLOATING POINT NUMBERS POINTED TO BY (H,L) AND (H,B).
;       THE ZERO FLIP-FLOP IS SET IF CHAR(H,L) EQUALS
;       CHAR(H,B).  IF CHAR(H,L) IS LESS THAN CHAR(H,B) THEN
;       THE CARRY BIT WILL BE SET.
;
;   REGISTERS ON EXIT:
;
;       A = CHARACTERISTIC OF (H,L) WITH SIGN EXTENDED
;       E = CHARACTERISTIC OF (H,B) WITH SIGN EXTENDED
;       D = A
```

```
                        ;       B,C,H,L = SAME AS ON ENTRY
925B    CD 4C 92        CCMP:   CALL    CFCHE           ;FETCH CHARACTERTISTICS WITH SIGN
                                                        ;EXTENDED INTO A (CHAR(H,L)) AND E
925E    57                      MOV     D,A             ;(CHAR(H,B)) REGIS SAVE CHAR (H,L)
925F    93                      SUB     E               ;SUBTRACT E (CHAR(H,B))
9260    17                      RAL                     ;ROTATE SIGN BIT INTO CARRY BIT
9261    7A                      MOV     A,D             ;RESTORE A=CHAR(H,L)
9262    C9                      RET                     ;RETURN
                        ;
                        ;   ERROR RETURNS
                        ;
                        ;       THE FOLLOWING CODE IS USED TO RETURN VARIOUS
                        ;       ERROR CONDITIONS.  IN EACH CASE A FLOATING POINT
                        ;       NUMBER IS STORED IN  THE 4 WORDS POINTED TO BY (H,L)
                        ;       AND A FLAG IS STORED IN THE ACCUMULATOR.
                        ;
                        ;       CONDITION    FLAG    RESULT (+)          RESULT (-)
                        ;
                        ;       UNDERFLOW     377    000 000 000 100     000 000 000 300
                        ;       OVERFLOW      177    377 377 377 077     377 377 377 277
                        ;       INDEFINITE    077    377 377 377 077     377 377 377 277
                        ;       NORMAL        000    XXX XXX XXX XXX     XXX XXX XXX XXX
                        ;       NORMAL ZERO   000    000 000 000 100     (ALWAYS RETURNS +0)
                        ;
                        ;   ENTRY POINTS:
                        ;
                        ;       WUND - WRITE UNDERFLOW
                        ;       WOVR - WRITE OVERFLOW
                        ;       WIND - WRITE INDEFINITE
                        ;       WZER - WRITE NORMAL ZERO
                        ;
1                       WFLT    MACRO   VMANT,VCHAR,VFLAG,LABEL  ;WRITE FLOATING NUMBER
1
1                               MVI     D,VCHAR         ;LOAD CHARACTERISTIC INTO D REGISTER
1                               CALL    WCHAR           ;WRITE CHARACTERISTIC
1                       LABEL:: MVI     A,VMANT         ;LOAD MANTISSA VALUE
1                                                       ;WE ASSUME HERE THAT ALL BYTES OF
1                                                       ;MANTISSA ARE THE SAME
1                               CALL    WMANT           ;WRITE THE MANTISSA
1                               MVI     A,VFLAG         ;SET ACCUMULATOR TO FLAG
```

```
                1                       ORA     A               ;SET FLAGS PROPERLY
                1                       RET                     ;RETURN (WMANT RESTORED (H,L))
                                        ENDM
                                ;
9263            1               +WUND:  WFLT    0,100Q,377Q,UFLW1  ;WRITE UNDERFLOW
                1               +
9263    1 16 40                 +       MVI     D,00040H        ;LOAD CHARACTERISTIC INTO D REGISTER
9265    1 CD 9F 92              +       CALL    WCHAR           ;WRITE CHARACTERISTIC
9268    1 3E 00                 +UFLW1::MVI     A,00000H        ;LOAD MANTISSA VALUE
                1               +                               ;WE ASSUME HERE THAT ALL BYTES OF
                1               +                               ;MANTISSA ARE THE SAME
926A    1 CD 98 92              +       CALL    WMANT           ;WRITE THE MANTISSA
926D    1 3E FF                 +       MVI     A,000FFH        ;SET ACCUMULATOR TO FLAG
926F    1 B7                    +       ORA     A               ;SET FLAGS PROPERLY
9270    1 C9                    +       RET                     ;RETURN (WMANT RESTORED (H,L))
9271            1               +WOVR:  WFLT    377Q,77Q,177Q,OFLW1  ;WRITE OVERFLOW
                1               +
9271    1 16 3F                 +       MVI     D,0003FH        ;LOAD CHARACTERISTIC INTO D REGISTER
9273    1 CD 9F 92              +       CALL    WCHAR           ;WRITE CHARACTERISTIC
9276    1 3E FF                 +OFLW1::MVI     A,000FFH        ;LOAD MANTISSA VALUE
                1               +                               ;WE ASSUME HERE THAT ALL BYTES OF
                1               +                               ;MANTISSA ARE THE SAME
9278    1 CD 98 92              +       CALL    WMANT           ;WRITE THE MANTISSA
927B    1 3E 7F                 +       MVI     A,0007FH        ;SET ACCUMULATOR TO FLAG
927D    1 B7                    +       ORA     A               ;SET FLAGS PROPERLY
927E    1 C9                    +       RET                     ;RETURN (WMANT RESTORED (H,L))
927F            1               +WIND:  WFLT    377Q,77Q,77Q,INDF1  ;WRITE INDEFINITE
                1               +
927F    1 16 3F                 +       MVI     D,0003FH        ;LOAD CHARACTERISTIC INTO D REGISTER
9281    1 CD 9F 92              +       CALL    WCHAR           ;WRITE CHARACTERISTIC
9284    1 3E FF                 +INDF1::MVI     A,000FFH        ;LOAD MANTISSA VALUE
                1               +                               ;WE ASSUME HERE THAT ALL BYTES OF
                1               +                               ;MANTISSA ARE THE SAME
9286    1 CD 98 92              +       CALL    WMANT           ;WRITE THE MANTISSA
9289    1 3E 3F                 +       MVI     A,0003FH        ;SET ACCUMULATOR TO FLAG
928B    1 B7                    +       ORA     A               ;SET FLAGS PROPERLY
928C    1 C9                    +       RET                     ;RETURN (WMANT RESTORED (H,L))
                                ;
928D      2C                     WZER:   INR     L               ;WRITE NORMAL ZERO
928E      2C                             INR     L               ;
928F      2C                             INR     L               ;
```

```
9290    36 40              MVI     M,100Q          ;STORE CHARACTERISTIC FOR ZERO
9292    AF                 XRA     A               ;ZERO ACCUMULATOR
9293    CD 98 92           CALL    WMANT           ;STORE ZERO MANTISSA
9296    B7                 ORA     A               ;SET FLAGS PROPERLY
9297    C9                 RET                     ;RETURN

                   ;  ROUTINE TO WRITE MANTISSA FOR ERROR RETURNS

9298    2D         WMANT:  DCR     L               ;POINT LEAST SIGNIFICANT BYTE
                                                   ;OF MANTISSA
9299    77                 MOV     M,A             ;STORE LSBYTE OF MANTISSA
929A    2D                 DCR     L               ;POINT TO NEXT LEAST SIGNIFICANT BYTE
                                                   ;OF MANTISSA
929B    77                 MOV     M,A             ;STORE NLSBYTE OF MANTISSA
929C    2D                 DCR     L               ;POINT TO MOST SIGNIFICANT BYTE
                                                   ;OF MANTISSA
929D    77                 MOV     M,A             ;STORE MSBYTE OF MANTISSA
929E    C9                 RET                     ;RETURN (H,L) POINTS TO BEGINNING OF
                                                   ;FLOATING POINT RESULT

                   ;  ROUTINE TO WRITE CHARACTERTIC FOR ERROR RETURNS
                   ;  NOTE:  WE PRESERVE ORIGINAL MANTISSA SIGN
                   ;  ON ENTRY D CONTAINS NEW CHARACTERTISTIC TO BE STORED.

929F    2C         WCHAR:  INR     L               ;SET (H,L) TO POINT TO CHARACTERISTIC
92A0    2C                 INR     L               ;PART OF ABOVE
92A1    2C                 INR     L               ;PART OF ABOVE
92A2    7E                 MOV     A,M             ;LOAD CHARACTERISTIC A
                                                   ;AND MANTISSA SIGN
92A3    E6 80              ANI     200Q            ;JUST KEEP MANTISSA SIGN
92A5    B2                 ORA     D               ;OR IN NEW CHARACTERISTIC
92A6    77                 MOV     M,A             ;STORE IT BACK
92A7    C9                 RET                     ;RETURN WITH (H,L) POINT TO
                                                   ;CHARACTERISTIC OF RESULT
                                                   ;SOMEONE ELSE WILL FIX UP (H,L)

                   ;    SUBROUTINE INDFC
                   ;
                   ;        THIS ROUTINE WRITES A FLOATING INDEFINITE, SETS
                   ;        THIS WRITES WRITES A FLOATING POINT INDEFINITE
                   ;        AT (H,C), SETS THE CONDITION FLAG AND RETURNS
                   ;
```

```
92A8  5D          INDFC:  MOV     E,L             ;SAVE LPTR IN E
92A9  69                  MOV     L,C             ;SET L=CPTR SO (H,L)-ADDR OF RESULT
92AA  CD 7F 92            CALL    WIND            ;WRITE INDEFINITE
92AD  6B                  MOV     L,E             ;RESTORE L=LPTR
92AE  C9                  RET                     ;RETURN

                  ;
                  ;   SUBROUTINE WZERC
                  ;
                  ;       THIS ROUTINE WRITES A NORMAL FLAOTING POINT ZERO
                  ;       AT (H,C), SETS THE CONDITION FLAG AND RETURNS
                  ;
92AF  5D          WZERC:  MOV     E,L             ;SAVE LPTR IN E
92B0  69                  MOV     L,C             ;SETL=CPTR SO (H,L)=ADDR OF RESULT
92B1  CD 8D 92            CALL    WZER            ;WRITE NORMAL ZERO
92B4  6B                  MOV     L,E             ;RESTORE L=LPTR
92B5  C9                  RET                     ;RETURN
                  ;
                  ;   SUBROUTINE INCR
                  ;
                  ;       THIS SUBROUTINE INCREMENTS THE CHARACTERISTIC
                  ;       OF THE FLOATING POINT NUMBER POINTED TO BY (H,L).
                  ;       WE TEST FOR OVERFLOW AND SET APPROPRIATE FLAG.
                  ;       (SEE ERRROR RETURNS).
                  ;
                  ;   REGISTERS ON EXIT:
                  ;
                  ;        A = CONDITION FLAG (SEE ERROR RETURNS)
                  ;        D = CLOBBERED
                  ;        B,C,H,L = SAME AS ON ENTRY
                  ;
92B6  CD 41 92    INCR:   CALL    GCHAR           ;GET CHAR WITH SIGN EXTENDED
92B9  FE 3F               CPI     MAXCH           ;COMPARE WITH MAX CHAR PERMITTED
92BB  CA 76 92            JZ      OFLW1           ;INCREMENT WOULD CAUSE OVERFLOW
92BE  57                  MOV D,A               ;/SAVE IT IN D
92BF  14                  INR D                 ;/INCREMENT IT
92C0  C3 C6 92            JMP     INCR2           ;JUMP AROUND ALTERNATE ENTRY POINT
92C3  2C          INCR3:  INR     L               ;COME HERE TO STORE CHARACTERISTIC
92C4  2C                  INR     L               ;POINT (H,L) TO CHAR
92C5  2C                  INR     L               ;POINT (H,L) TO CHAR
```

```
92C6    3E 7F          INCR2:   MVI A,177Q
92C8    A2                      ANA D                ;/KILL SIGN BIT
92C9    57                      MOV D,A              ;/BACK TO D
92CA    7E                      MOV A,M              ;/NOW SIGN IT
92CB    E6 80                   ANI 200Q             ;/GET MANTISSA SIGN
92CD    B2                      ORA D                ;/PUT TOGETHER
92CE    77                      MOV M,A              ;/STORE IT BACK
92CF    2D                      DCR L                ;/NOW BACK TO BASE
92D0    2D                      DCR L                ;/***TP
92D1    2D                      DCR L
92D2    AF             SCCFG:    XRA    A                    ;SET SUCCESS FLAG
92D3    C9                      RET
                       ;
                       ;    SUBROUTINE DECR
                       ;
                       ;        THIS SUBROUTINE DECREMENTS THE CHARACTERISTIC
                       ;        OF THE FLOATING POINT NUMBER POINTED TO BY (H,L).
                       ;        WE TEST FOR UNDERFLOW AND SET APPROPRIATE FLAG.
                       ;        (SEE ERRROR RETURNS).
                       ;
                       ;    REGISTERS ON EXIT:
                       ;
                       ;         A = CONDITION FLAG (SEE ERROR RETURNS)
                       ;         D = CLOBBERED
                       ;         B,C,H,L = SAME AS ON ENTRY
                       ;
92D4    CD 41 92       DECR:    CALL    GCHAR        ;GET CHAR WITH SIGN EXTENDED
92D7    FE C0                   CPI     MINCH        ;COMPARE WITH MIN CHAR PERMITTED
92D9    CA 68 92                JZ      UFLW1        ;DECREMENT WOULD CAUSE UNDERFLOW
92DC    57                      MOV     D,A          ;SAVE CHARACTERSTIC IN D
92DD    15                      DCR     D            ;DECREMENT CHARACTERISTIC
92DE    C3 C6 92                JMP     INCR2        ;GO STORE IT BACK
                       ;
                       ;                            SUBROUTINE AORS
                       ;                            RETURN S 1 IF BASE  6
                       ;                            HAS A 1 IN MSB
92E1    5D             AORS:    MOV E,L                ;SAVE BASE
92E2    69                      MOV L,C                ;BASE  6 TO L
92E3    7E                      MOV A,M                ;LOAD IT
92E4    B7                      ORA A                  ;SET FLAGS
92E5    6B                      MOV L,E                ;RESTORE BASE
```

```
92E6    C9                      RET
                        ;
                        ;
                        ;
                        ;
92E7    5D              TSTR:   MOV E,L         SUBROUTINE TSTR
92E8    69                      MOV L,C         CHECKS C PTR TO SEE IF
92E9    16 02                   MVI D,2         NLSB 1
92EB    7E                      MOV A,M         RETURNS Z 1 IF NOT
92EC    6B                      MOV L,E         DESTROYS E,D
92ED    A2                      ANA D             ;SAVE BASE
92EE    C9                      RET               ;C PTR TO L
                                                  ;MASK TO D
                        ;                         ;LOAD VALUE
                        ;                         ;RESTORE BASE
                        ;                         ;AND VALUE WITH MASK
92EF    5D              ACPR:   MOV E,L
92F0    69                      MOV L,C         SUBROUTINE ACPR
92F1    77                      MOV M,A         STORES A IN LOCATION OF CPTR
92F2    6B                      MOV L,E         LPTR IN E
92F3    C9                      RET               ;SAVE LPTR
                                                  ;CPTR TO L
                        ;                         ;STORE A
                        ;                         ;RESTORE BASE
92F4    7E              DCMP:   MOV A,M
92F5    5D                      MOV E,L         SUBROUTINE DCMP
92F6    68                      MOV L,B         COMPARES TWO DOUBLE LENGTH
92F7    BE                      CMP M           WORDS
92F8    6B                      MOV L,E           ;NUM MANTA TO A
92F9    C0                      RNZ               ;SAVE BASE IN E
92FA    2C                      INR L             ;BASE 3 TO L
92FB    7E                      MOV A,M           ;COMPARE WITH DEN MANTA
92FC    68                      MOV L,B           ;RETURN BASE TO L
92FD    2C                      INR L             ;RETURN IF NOT THE SAME
92FE    BE                      CMP M             ;L TO NUM MANTB
92FF    6B                      MOV L,E           ;LOAD IT
9300    C0                      RNZ               ;DEN MANTB ADD TO L
9301    2C                      INR L             ;BASE  4 TO L
9302    2C                      INR L
9303    7E                      MOV A,M           ;/***TP EXTENSION
9304    68                      MOV L,B           ;/NOW CHECK BYTE 3

                                                  ;/GET FOR COMPARE
```

```
9305    2C                      INR L
9306    2C                      INR L
9307    BE                      CMP M           ;/BYTE 3 NOW
9308    6B                      MOV L,E         ;/COMPARE
9309    C9                      RET             ;/***TP - ALL DONE

                        ;
                        ;
                        ;
                        ;
                        ;
930A    CD E9 91        ENT2:   CALL DLST       SUBROUTINE DIVC
930D    DA 17 93                JC  OVER        PERFORMS ONE CYCLE OF DOUBLE
9310    CD F4 92        ENT1:   CALL DCMP       PRECISION FLOATING PT DIVIDE
9313    D2 17 93                JNC OVER        ENTER AT ENT1 ON FIRST CYCLE
9316    C9                      RET             ENTER AT ENT2 ALL THEREAFTER
9317    CD 26 92        OVER:   CALL DSUB       ;SHIFT MOVING DIVIDEND
931A    5D                      MOV E,L         ;IF CARRY 1,NUM.GT.D
931B    69                      MOV L,C         ;COMPARE NUM WITH DEN
931C    2C                      INR L           ;IF CARRY NOT SET,NUM.GE.DEN
931D    2C                      INR L
931E    7E                      MOV A,M         ;CALL DOUBLE SUBTRACT
931F    C6 01                   ADI 1           ;SAVE BASE IN E
9321    77                      MOV M,A         ;BASE  6 TO L
9322    6B                      MOV L,E         ;BASE  7 TO L
9323    C9                      RET             ;/***TP

                        ;
                        ;
                        ;
9324    16 04           LXFR:   MVI D,4         SUBROUTINE LXFR
9326    69              REP5:   MOV L,C         MOVES CPTR TO EPTR
9327    7E                      MOV A,M         MOVES 3 WORDS IF ENTER AT LXFR
9328    6B                      MOV L,E         ;/MOVE 4 WORDS
9329    77                      MOV M,A         ;CPTR TO L
932A    0C                      INR C           ; CPTR> TO A
932B    1C                      INR E           ;EPTR TO L
932C    15                      DCR D
932D    C2 26 93                JNZ REP5        ;/INCREMENT C
9330    7B                      MOV A,E         ;/INCREMENT E TO NEXT
9331    D6 04                   SUI 4           ;/TEST FOR DONE
9333    5F                      MOV E,A         ;/GO FOR FOR TILL D=0
9334    79                      MOV A,C         ;/NOW RESET C AND E
                                                ;/RESET BACK BY 4
                                                ;/PUT BACK IN E
                                                ;/NOW RESET C
```

```
9335    D6 04                   SUI 4               ;/BY 4
9337    4F                      MOV C,A             ;/BACK TO C
9338    C9                      RET                 ;/DONE
                        ;
                        ;   SUBROUTINE LDCP
                        ;
                        ;       THIS SUBROUTINE COMPUTES THE CHARACTERISTIC
                        ;       FOR THE FLOATING DIVIDE ROUTINE
                        ;
                        ;   REGISTERS ON EXIT:
                        ;
                        ;       A = CONDITION FLAG (SEE ERROR RETURNS)
                        ;       D,E = GARBAGE
                        ;       B,C,H,L = SAME AS ON ENTRY
                        ;
                        ;   REGISTERS ON ENTRY:
                        ;
                        ;       (H,B) = ADDRESS OFF DIVISOR
                        ;       (H,C) = ADDRESS OF QUOTIENT
                        ;       (H,L) = ADDRESS OF DIVIDEND
9339    CD 4C 92    LDCP:   CALL    CFCHE       ;SET E=CHAR(H,B), A=CHAR(H,L)
933C    93                  SUB     E           ;SUBTRACT TO GET NEW CHARACTERISTIC
933D    C3 44 93            JMP     CCHK        ;GO CHECK FOR OVER/UNDERFLOW
                                                ;AND STORE CHARACTERTISTIC
                        ;
                        ;   SUBROUTINE LMCP
                        ;
                        ;       THIS SUBROUTINE COMPUTES THE CHARACTERISTIC
                        ;       FOR THE FLOATING MULTIPLY ROUTINE.
                        ;
                        ;   REGISTERS ON EXIT:
                        ;
                        ;       A = CONDITION FLAG (SEE ERROR RETURNS)
                        ;       D,E = GARBAGE
                        ;       B,C,H,L = SAME AS ON ENTRY
                        ;
                        ;   REGISTERS ON ENTRY:
                        ;
                        ;       (H,B) = ADDRESS OFF MULTIPLICAND
```

```
                        ;       (H,C) = ADDRESS OF PRODUCT
                        ;       (H,L) = ADDRESS OF MULTIPLIER
                        ;
9340    CD 4C 92        LMCP:   CALL    CFCHE           ;SET E=CHAR(H,B), A=CHAR(H,L)
9343    83                      ADD     E               ;ADD TO GET NEW CHARACTERISTIC
                                                        ;NOW FALL INTO THE ROUTINE
                                                        ;WHICH CHECKS FOR OVER/UNDERFLOW
                                                        ;AND STORE CHARACTERTISTIC
                        ;
                        ;   SBUROUTINE CCHK
                        ;
                        ;       THIS SUBROUTINE CHECKS A CHARACTERISTIC IN
                        ;       THE ACCUMULATOR FOR OVERFLOW OR UNDERFLOW.
                        ;       IT THEN STORES THE CHARACTERISTIC, PRESERVING
                        ;       THE PREVIOUSLY COMPUTED MANTISSA SIGN.
                        ;
                        ;   REGISTERS ON ENTRY:
                        ;
                        ;       (H,L) = ADDRESS OF ONE OPERAND
                        ;       (H,B) = ADDRESS OF OTHER OPERAND
                        ;       (H,C) = ADDRESS OF RESULT
                        ;       A     = NEW CHARACTERISTIC OF  RESULT
                        ;
                        ;   REGISTERS ON EXIT:
                        ;
                        ;       A = CONDITION FLAG (SEE ERROR RETURNS)
                        ;       D,E = GARBAGE
                        ;       B,C,H,L = SAME AS ON ENTRY
                        ;
9344                    CCHK:                           ;ENTER HERE TO CHECK CHARACTERISTIC
9344    FE 40                   CPI     100Q            ;CHECK FOR 0 TO +63
9346    DA 53 93                JC      STORC           ;JUMP IF OKAY
9349    FE 80                   CPI     200Q            ;CHECK FOR +64 TO +127
934B    DA 5B 93                JC      OFLWC           ;JUMP IF OVERFLOW
934E    FE C0                   CPI     300Q            ;CHECK FOR -128 TO -65
9350    DA 62 93                JC      UFLWC           ;JUMP IF UNDERFLOW
9353    5D              STORC:  MOV     E,L             ;SAVE L IN E
9354    69                      MOV     L,C             ;LET L POINT TO RESULT
9355    57                      MOV     D,A             ;SAVE CHARACTERISTIC IN D
9356    CD C3 92                CALL    INCR3           ;STORE CHARACTERISTIC
```

```
9359    6B                      MOV     L,E             ;RESTORE L
935A    C9                      RET                     ;RETURN
                        ;
                        ;   SUBROUTINE OFLWC
                        ;
                        ;       THIS ROUTINE WRITES A FLOATING POINT OVERFLOW AT (H,C)
                        ;       SETS THE CONDITION FLAG, AND RETURNS.
                        ;
935B    5D              OFLWC:  MOV     E,L             ;SAVE L IN E
935C    69                      MOV     L,C             ;SET L=CPTR, SO (H,L)=ADDR OF RESULT
935D    CD 71 92                CALL    WOVR            ;WRITE OUT OVERFLOW
9360    6B                      MOV     L,E             ;RESTORE L
9361    C9                      RET                     ;RETURN
                        ;
                        ;   SUBROUTINE UFLWC
                        ;
                        ;       THIS ROUTINE WRITES A FLOATING POINT UNDERFLOW AT (H,C)
                        ;       SETS THE CONDITION FLAG, AND RETURNS.
                        ;
9362    5D              UFLWC:  MOV     E,L             ;SAVE L IN E
9363    69                      MOV     L,C             ;SET L=CPTR, SO (H,L)=ADDR OF RESULT
9364    CD 63 92                CALL    WUND            ;WRITE OUT UNDEFLOW
9367    6B                      MOV     L,E             ;RESTORE L
9368    C9                      RET                     ;RETURN
                        ;
                        ;   SUBROUTINE CSIGN
                        ;
                        ;       THIS SUBROUTINE COMPUTES AND STORE THE MANTISSA
                        ;       SIGN FOR THE FLOATING MULTIPLY AND DIVIDE ROUTINES
                        ;
                        ;   REGISTERS ON ENTRY:
                        ;
                        ;       (H,L) = ADDRESS OF ONE OPERAND
                        ;       (H,B) = ADDRESS OF OTHER OPERAND
                        ;       (H,C) = ADDRESS OF RESULT
                        ;
                        ;   REGISTERS ON EXIT:
                        ;
                        ;       A,D,E = GARBAGE
                        ;       B,C,H,L = SAME AS ON ENTRY
```

```
                                ;
9369    CD 79 93        CSIGN:  CALL    MSFH            ;SET A=SIGN(H,L), E=SIGN(H,B)
936C    AB                      XRA     E               ;EXCLUSIVE OR SIGNS TO GET NEW SIGN
936D    CD 71 93                CALL    CSTR            ;STORE SIGN INTO RESULT
9370    C9                      RET                     ;RETURN
                                ;
                                ;
                                ;                       SUBROUTINE CSTR
                                ;                       STORES VALUE IN A IN
                                ;                       CPTR 2
                                ;                       PUTS LPTR IN E
9371    5D              CSTR:   MOV E,L                 ;SAVE LPTR IN E
9372    69                      MOV L,C                 ;CPTR TO L
9373    2C                      INR L                   ;CPTR 2
9374    2C                      INR L                   ;TO L
9375    2C                      INR L                   ;/***TP
9376    77                      MOV M,A                 ;STORE ANSWER
9377    6B                      MOV L,E                 ;LPTR BACK TO L
9378    C9                      RET
                                ;
                                ;   SUBROUTINE MSFH
                                ;
                                ;       THIS SUBROUTINE FETCHES THE SIGNS OF THE MANTISSAS
                                ;       OF THE FLOATING POINT NUMBERS POINTED TO BY (H,L)
                                ;       AND (H,B) INTO THE A AND E REGISTERS RESPECTIVELY.
                                ;
                                ;   REGISTERS ON EXIT:
                                ;
                                ;       A = SIGN  OF MANTISSA OF (H,L)
                                ;       E = SIGN OF MANTISSA OF (H,B)
                                ;       B,C,D,H,L = SAME AS ON ENTRY
                                ;
9379    5D              MSFH:   MOV E,L                 ;SAVE LPTR
937A    68                      MOV L,B                 ;BPTR TO L
937B    2C                      INR L                   ;BPTR 2
937C    2C                      INR L                   ;/***TP
937D    2C                      INR L                   ;TO L
937E    7E                      MOV A,M                 ; BPTR 2>TO A
937F    E6 80                   ANI 128                 ;SAVE MANT SIGN
9381    6B                      MOV L,E                 ;LPTR BACK TO L
```

```
9382  5F                MOV E,A          ;STORE BPTR MANT SIGN
9383  2C                INR L            ;LPTR 2
9384  2C                INR L            ;/***TP
9385  2C                INR L            ;TO L
9386  7E                MOV A,M          ; LPTR 2>TO A
9387  E6 80             ANI 128          ;SAVE LPTR MANT SIGN
9389  2D                DCR L            ;LPTR BACK
938A  2D                DCR L            ;TO L
938B  2D                DCR L            ;/***TP
938C  C9                RET
                      ;                  SUBROUTINE BCTL
                      ;                  MOVES BPTR CHAR TO LPTR CHAR
                      ;                  DESTROYSE
938D  5D              BCTL:  MOV E,L     ;LPTR TO E
938E  68                MOV L,B          ;BPTR TO L
938F  2C                INR L            ;BPTR  2
9390  2C                INR L            ;/***TP
9391  2C                INR L            ;TO L
9392  7E                MOV A,M          ;BPTR CHAR TO A
9393  6B                MOV L,E          ;LPTR TO L
9394  2C                INR L            ;LPTR  2
9395  2C                INR L            ;TO L
9396  2C                INR L            ;/***TP
9397  77                MOV M,A          ;STORE BPTR CHAR IN LPTR CHAR
9398  6B                MOV L,E          ;LPTR TO L
9399  C9                RET
                      ;
                      ;*****************************************************
                      ;       //// 5 DIGIT FLOATING PT. OUTPUT
                      ;*****************************************************
                      ;
                      ;
                      ;       *******ROUTINE TO CONVERT FLOATING PT.
                      ;       ***NUMBERS TO ASCII AND OUTPUT THEM VIA A SUBROUTINE
                      ;       ***CALLED OUTR  -  NOTE: THIS IS CURRENTLY SET
                      ;       ***TO ODT'S OUTPUT ROUTINE
                      ;
```

```
939A  CD DA 91    CVRT:   CALL   ZCHK            ;CHECK FOR NEW ZERO
939D  C2 AF 93            JNZ    NNZRO           ;NOT ZERO
93A0  0C                  INR    C               ;IT WAS, OFFSET C BY 2
93A1  0C                  INR    C
93A2  69                  MOV    L,C
93A3  CD 8D 92            CALL   WZER            ;WRITE ZERO
93A6  2C                  INR    L               ;PNT TO DECIMAL EXPONENT
93A7  2C                  INR    L
93A8  2C                  INR    L
93A9  2C                  INR    L
93AA  AF                  XRA    A               ;SET IT TO ZERO
93AB  77                  MOV    M,A
93AC  C3 0E 94            JMP    MDSKP           ;OUTPUT IT
93AF  56          NNZRO:  MOV    D,M             ;/GET THE NUMBER TO CONVERT
93B0  2C                  INR L
93B1  46                  MOV B,M
93B2  2C                  INR L
93B3  5E                  MOV E,M
93B4  2C                  INR L                  ;/4 WORD***TP
93B5  7E                  MOV A,M                ;/***TP
93B6  0C                  INR C                  ;/OFFSET SCRATCH POINTER BY 2
93B7  0C                  INR C
93B8  69                  MOV L,C                ;/L NOT NEEDED ANY MORE
93B9  72                  MOV M,D                ;/SAVE NUMBER IN SCRATCH
93BA  2C                  INR L
93BB  70                  MOV M,B
93BC  2C                  INR L
93BD  73                  MOV M,E                ;/***TP
93BE  2C                  INR L                  ;/***TP
93BF  47                  MOV B,A                ;/SAVE COPY OF CHAR & SIGN
93C0  E6 7F               ANI 177Q               ;GET ONLY CHAR.
93C2  77                  MOV M,A                ;/SAVE ABS(NUMBER)
93C3  FE 40               CPI 100Q               ;CK FOR ZERO
93C5  CA CC 93            JZ  NZRO
93C8  D6 01               SUI 1                  ;/GET SIGN OF DEC. EXP
93CA  E6 40               ANI 100Q               ;/GET SIGN OF CHAR.
93CC  07          NZRO:   RLC                    ;MOVE IT TO SIGN POSITION
93CD  2C                  INR L                  ;/MOVE TO DECIMAL EXP.
93CE  77                  MOV M,A                ;/SAVE SIGN OF EXP.
93CF  78                  MOV A,B                ;/GET MANT. SIGN BACK
93D0  CD 90 94            CALL SIGN              ;/OUTPUT SIGN
```

```
93D3  2E 14              MVI L,(TEN5 AND 377Q)  ;/TRY MULT. OR DIV. BY 100000 FIRST
93D5  CD F1 94           CALL COPT              ;/MAKE A COPY IN RAM
93D8  CD 9D 94     TST8: CALL GCHR              ;/GET CHAR. OF NUMBER
93DB  47                 MOV B,A                ;/SAVE A COPY
93DC  E6 40              ANI 100Q               ;/GET ABSOLUTE VALUE OF CHAR
93DE  78                 MOV A,B                ;/INCASE PLUS
93DF  CA E5 93           JZ  GOTV               ;/ALREADY PLUS
93E2  3E 80              MVI A,200Q             ;/MAKE MINUS INTO PLUS
93E4  90                 SUB B                  ;/PLUS=200B-CHAR
93E5  FE 12        GOTV: CPI 22Q                ;/TEST FOR USE OF 100000
93E7  FA F3 93           JM  TRY1               ;/WONT GO
93EA  CD A3 94           CALL MORD              ;/WILL GO SO DO IT
93ED  C6 05              ADI 5                  ;/INCREMENT DEC. EXPONENT BY 5
93EF  77                 MOV M,A                ;/UPDATE MEM
93F0  C3 D8 93           JMP TST8               ;/GO TRY AGAIN
93F3  2E 18        TRY1: MVI L,(TEN AND 377Q)   ;/NOW USE JUST TEN
93F5  CD F1 94           CALL COPT              ;/PUT IT IN RAM
93F8  CD 9D 94     TST1: CALL GCHR              ;/GET CHARACTERISTIC
93FB  FE 01              CPI 1                  ;/MUST GET IN RANGE 1 TO 6
93FD  F2 09 94           JP  OK1                ;/ATLEAST ITS 1 OR BIGGER
9400  CD A3 94     MDGN: CALL MORD              ;/MUST MUL OF DIV BY 10
9403  C6 01              ADI 1                  ;/INCREMENT DECIMAL EXP.
9405  77                 MOV M,A                ;/UPDATE MEM
9406  C3 F8 93           JMP TST1               ;/NOW TRY AGAIN
9409  FE 07        OK1:  CPI 7                  ;/TEST FOR LESS THAN 7
940B  F2 00 94           JP  MDGN               ;/NOPE - 7 OR GREATER
940E  69           MDSKP:  MOV    L,C                ;/SET UP DIGIT COUNT
940F  2D                 DCR L
9410  2D                 DCR L                  ;/IN 1ST WORD OF SCRATCH
9411  36 05              MVI M,5                ;/5 DIGITS
9413  5F                 MOV E,A                ;/SAVE CHAR. AS LEFT SHIFT COUNT
9414  CD 76 94           CALL LSFT              ;/SHIFT LEFT PROPER NUMBER
9417  FE 0A              CPI 12Q                ;/TEST FOR 2 DIGITS HERE
9419  F2 C9 94           JP  TWOD               ;/JMP IF 2 DIGITS TO OUTPUT
941C  CD 3A 94           CALL DIGO              ;/OUTPUT FIRST DIGIT
941F  CD 4E 94     POPD: CALL MULTT             ;/MULTIPLY THE NUMBER BY 10
9422  CD 3A 94     INPOP: CALL DIGO             ;/PRINT DIGIT IN A
9425  C2 1F 94           JNZ POPD               ;/MORE DIGITS?
9428  3E C5              MVI A,305Q             ;/NO SO PRINT E
942A  CD FD 97           CALL OUTR              ;/BASIC CALL TO OUTPUT
942D  CD BE 94           CALL GETEX             ;/GET DECIMAL EXP
```

```
9430  47                      MOV B,A         ;/SAVE A COPY
9431  CD 90 94                CALL SIGN       ;/OUTPUT SIGN
9434  78                      MOV A,B         ;/GET EXP BACK
9435  E6 3F                   ANI 77Q         ;/GET GOOD BITS
9437  CD E0 94                CALL CTWO       ;/GO CONVERT 2 DIGITS
943A  C6 B0             DIGO: ADI 260Q        ;/MAKE A INTO ASCII
943C  CD FD 97                CALL OUTR       ;/OUTPUT DIGIT
943F  69                      MOV L,C         ;/GET DIGIT COUNT
9440  2D                      DCR L           ;/BACK UP TO DIGIT COUNT
9441  2D                      DCR L
9442  7E                      MOV A,M         ;/TEST FOR DECIMAL PT
9443  FE 05                   CPI 5           ;/PRINT . AFTER 1ST DIGIT
9445  3E AE                   MVI A,256Q      ;/JUST IN CASE
9447  CC FD 97                CZ  OUTR        ;/OUTPUT . IF 1ST DIGIT
944A  56                      MOV D,M         ;/NOW DECREMENT DIGIT COUNT
944B  15                      DCR D
944C  72                      MOV M,D         ;/UPDATE MEM AND LEAVE FLOPS SET
944D  C9                      RET             ;/SERVES AS TERM FOR DIGO & CVRT
944E  1E 01             MULTT: MVI E,1        ;/MULT. BY 10 (START WITH X2)
9450  CD 76 94                CALL LSFT       ;/LEFT SHIFT 1 = X2
9453  69                      MOV L,C         ;/SAVE X2 IN "RESULT"
9454  2D                      DCR L           ;/SET TO TOP OF NUMBER
9455  79                      MOV A,C         ;/SET C TO RESULT
9456  C6 09                   ADI 11Q
9458  4F                      MOV C,A         ;/NOW C SET RIGHT
9459  7C                      MOV A,H         ;/SHOW RAM TO RAM TRANSFER
945A  CD FF 94                CALL COPY       ;/SAVE X2 FINALLY
945D  79                      MOV A,C         ;/MUST RESET C
945E  D6 09                   SUI 11Q         ;/BACK TO NORMAL
9460  4F                      MOV C,A
9461  1E 02                   MVI E,2         ;/NOW GET (X2)X4=X8
9463  69                      MOV L,C         ;/BUT MUST SAVE OVERFLOW
9464  2D                      DCR L
9465  CD 7A 94                CALL TLP2       ;/GET X8
9468  69                      MOV L,C         ;/SET UP TO CALL DADD
9469  79                      MOV A,C         ;/SET B TO X2
946A  C6 0A                   ADI 12Q         ;/TO X2
946C  47                      MOV B,A
946D  CD 06 92                CALL DADD       ;/ADD TWO LOW WORDS
9470  2D                      DCR L           ;/BACK UP TO OVERFLOW
9471  7E                      MOV A,M         ;/GET IT
```

```
9472  68                    MOV L,B     ;/NOW SET TO X2 OVERFLOW
9473  2D                    DCR L       ;/ITS AT B-1
9474  8E                    ADC M       ;/ADD WITH CARRY - CARRY WAS PRESERVED
9475  C9                    RET         ;/ALL DONE, RETURN OVERFLOW IN A
9476  69              LSFT: MOV L,C     ;/SET PTR FOR LEFT SHIFT OF NUMBER
9477  2D                    DCR L       ;/BACK UP TO OVERFLOW
9478  AF                    XRA A       ;/OVERFLOW=0 1ST TIME
9479  77              TLOOP: MOV M,A    ;/SAVE OVERFLOW
947A  1D              TLP2: DCR E       ;/TEST FOR DONE
947B  F8                    RM          ;/DONE WHEN E MINUS
947C  2C                    INR L       ;/MOVE TO LOW
947D  2C                    INR L
947E  2C                    INR L       ;/***TP EXTENSION
947F  7E                    MOV A,M     ;/SHIFT LEFT 4 BYTES
9480  17                    RAL
9481  77                    MOV M,A     ;/PUT BACK
9482  2D                    DCR L       ;/***TP - ALL DONE
9483  7E                    MOV A,M     ;/GET LOW
9484  17                    RAL         ;/SHIFT LEFT 1
9485  77                    MOV M,A     ;/RESTORE IT
9486  2D                    DCR L       ;/BACK UP TO HIGH
9487  7E                    MOV A,M     ;/GET HIGH
9488  17                    RAL         ;/SHIFT IT LEFT WITH CARRY
9489  77                    MOV M,A     ;/PUT IT BACK
948A  2D                    DCR L       ;/BACK UP TO OVERFLOW
948B  7E                    MOV A,M     ;/GET OVERFLOW
948C  17                    RAL         ;/SHIFT IT LEFT
948D  C3 79 94              JMP TLOOP   ;/GO FOR MORE
9490  E6 80           SIGN: ANI 200Q    ;/GET SIGN BIT
9492  3E A0                 MVI A,240Q  ;/SPACE INSTEAD OF PLUS
9494  CA 99 94              JZ  PLSV    ;/TEST FOR +
9497  3E AD                 MVI A,255Q  ;/NEGATIVE
9499  CD FD 97        PLSV: CALL OUTR   ;/OUTPUT SIGN
949C  C9                    RET
949D  69              GCHR: MOV L,C     ;/GET CHARCTERISTIC
949E  2C              GETA: INR L       ;/MOVE TO IT
949F  2C                    INR L
94A0  2C                    INR L       ;/***TP
94A1  7E                    MOV A,M     ;/FETCH INTO A
94A2  C9                    RET         ;/DONE
94A3  CD BE 94        MORD: CALL GETEX  ;/MUL OR DIV DEPENDING ON EXP
```

```
94A6  5F                  MOV E,A      ;/SAVE DECIMAL EXP
94A7  45                  MOV B,L      ;/SET UP TO MULT OR DIV
94A8  04                  INR B        ;/NOW BOP POINTER SET
94A9  69                  MOV L,C      ;/L POINTS TO NUMBER TO CONVERT
94AA  79                  MOV A,C      ;/POINT C AT "RESULT" AREA
94AB  C6 09               ADI 11Q      ;/IN SCRATCH
94AD  4F                  MOV C,A      ;/NOW C SET RIGHT
94AE  7B                  MOV A,E      ;/NOW TEST FOR MUL
94AF  E6 80               ANI 200Q     ;/TEST NEGATIVE DEC. EXP.
94B1  CA C3 94            JZ  DIVIT    ;/IF EXP IS + THEN DIVIDE
94B4  CD 5F 91            CALL LMUL    ;/MULT.
94B7  79            FINUP: MOV A,C      ;/SAVE LOC. OF RESULT
94B8  4D                  MOV C,L      ;/C=LOC OF NUMBER (IT WAS DESTROYED)
94B9  6F                  MOV L,A      ;/SET L TO LOC. OF RESUTL
94BA  7C                  MOV A,H      ;/SHOW RAM TO RAM TRANSFER
94BB  CD FF 94            CALL COPY    ;/MOVE RESULT TO NUMBER
94BE  69            GETEX: MOV L,C      ;/NOW GET DECIMAL EXP
94BF  2C                  INR L
94C0  C3 9E 94            JMP GETA     ;/USE PART OF GCHR
94C3  CD 00 90      DIVIT: CALL LDIV    ;/DIVIDE
94C6  C3 B7 94            JMP FINUP
94C9  CD E0 94      TWOD:  CALL CTWO    ;/CONVERT TO 2 DIGITS
94CC  47                  MOV B,A      ;/SAVE ONES DIGIT
94CD  CD BE 94            CALL GETEX   ;/GET DECIMAL EXP
94D0  5F                  MOV E,A      ;/SAVE A COPY
94D1  E6 80               ANI 200Q     ;/TEST FOR NEGATIVE
94D3  CA DC 94            JZ  ADD1     ;/BUMP EXP BY 1 SINCE 2 DIGITS
94D6  1D                  DCR E        ;/DECREMENT NEGATIVE EXP SINCE 2 DIGITS
94D7  73            FINIT: MOV M,E      ;/RESTORE EXP WITH NEW VALUE
94D8  78                  MOV A,B      ;/NOW DO 2ND DIGIT
94D9  C3 22 94            JMP INPOP    ;/GO OUT 2ND AND REST FO DIGITS
94DC  1C            ADD1:  INR E        ;/COMPENSATE FOR 2 DIGITS
94DD  C3 D7 94            JMP FINIT
94E0  1E FF         CTWO:  MVI E,377Q   ;/CONVERT 2 DIGIT BIN TO BCD
94E2  1C            LOOP:  INR E        ;/ADD UP TENS DIGIT
94E3  D6 0A               SUI 12Q      ;/SUBTRACT 10
94E5  F2 E2 94            JP  LOOP     ;/TIIL NEGATIVE RESULT
94E8  C6 0A               ADI 12Q      ;/RESTORE ONES DIGIT
94EA  47                  MOV B,A      ;/SAVE ONES DIGIT
94EB  7B                  MOV A,E      ;/GET TENS DIGIT
94EC  CD 3A 94            CALL DIGO    ;/OUTPUT IT
```

```
94EF  78                      MOV A,B           ;/SET A TO 2ND DIGIT
94F0  C9                      RET
94F1  79                COPT: MOV A,C           ;/COPY FROM 10 N TO RAM
94F2  C6 05                   ADI 5
94F4  4F                      MOV C,A           ;/SET C TO PLACE TO PUT
94F5  3E 95                   MVI A,(TEN5/256)
94F7  CD FF 94                CALL COPY         ;/COPY IT
94FA  79                      MOV A,C           ;/NOW RESET C
94FB  D6 05                   SUI 5
94FD  4F                      MOV C,A           ;/ITS RESET
94FE  C9                      RET
94FF  44                COPY: MOV B,H           ;/SAVE RAM H
9500  67                      MOV H,A           ;/SET TO SOURCE H
9501  7E                      MOV A,M           ;/GET 4 WORDS INTO THE REGS.
9502  2C                      INR L
9503  56                      MOV D,M
9504  2C                      INR L
9505  5E                      MOV E,M
9506  2C                      INR L
9507  6E                      MOV L,M           ;/LAST ONE ERASES L
9508  60                      MOV H,B           ;/SET TO DESTINATION RAM
9509  45                      MOV B,L           ;/SAVE 4TH WORD IN B
950A  69                      MOV L,C           ;/SET TO DESTINATION
950B  77                      MOV M,A           ;/SAVE FIRST WORD
950C  2C                      INR L
950D  7E                      MOV A,M           ;/SAVE THIS WORD IN A (INPUT SAVES C HERE
950E  72                      MOV M,D           ;/NOW PUT 2ND WORD
950F  2C                      INR L
9510  73                      MOV M,E
9511  2C                      INR L
9512  70                      MOV M,B           ;/ALL 4  COPIED NOW
9513  C9                      RET               ;/ALL DONE
                        ;
9514  C3 50 00 11     TEN5: DB 303Q,120Q,00,21Q  ;/303240(8) = 100000.
9518  A0 00 00 04     TEN:  DB 240Q,00,00,4Q     ;/12(8) = 10
                        ;
                        ;     SCRATCH MAP FOR I/O CONVERSION ROUTINES
                        ;
                        ;     RELATIVE TO (C+2)USE
                        ;     C-2               DIGIT COUNT
```

```
                                ;       C-1             OVERFLOW
                                ;       C               HIGH NUMBER - MANTISSA
                                ;       C+1             LOW NUMBER
                                ;       C+2             CHARACTERISTIC
                                ;       C+3             DECIMAL EXPONEXT (SIGN & MAG.)
                                ;       C+4             TEN**N
                                ;       C+5             TEN**N
                                ;       C+6             TEN**N
                                ;       C+7             RESULT OF MULT & DIV
                                ;       C+8             AND TEMP FOR X2
                                ;       C+9             "
                                ;       C+10            L FOR NUMBER TO GO INTO (INPUT ONLY)
                                ;       C+11            DIGIT JUST INPUT (INPUT ONLY)
                                ;
                                ;                       /*****BEGIN INPUT*************
                                ;
951C    37              ERR:    STC                     ;ERROR FLAG
951D    C9                      RET                     ;AND RETURN
                                ;
                                ;*****************************************************
                                ;       //// 4 1/2 DIGIT INPUT ROUTINE
                                ;*****************************************************
                                ;
                                ;                       /L POINTS TO WHERE TO PUT INPUT NUMBER
                                ;                       /C POINTS TO 13(10) WORDS OF SCRATCH
                                ;
951E    45              INPUT:  MOV B,L                 ;/SAVE ADDRESS WHERE DATA IS TO GO
951F    79                      MOV A,C                 ;/IN SCRATCH
9520    C6 0F                   ADI 17Q                 ;/COMPUTE LOC. IN SCRATCH
9522    6F                      MOV L,A
9523    70                      MOV M,B                 ;/PUT IT
9524    0C                      INR C                   ;/OFFSET SCRATCH POINTER
9525    0C                      INR C                   ;/BY 2
9526    CD 2D 96                CALL ZROIT              ;/ZERO NUMBER
9529    2C                      INR L                   ;/AND ZERO
952A    77                      MOV M,A                 ;/DECIMAL EXPONENT
952B    CD C9 95                CALL GNUM               ;/GET INTEGER PART OF NUM
952E    FE FE                   CPI 376Q                ;/TERM=.?
```

```
9530    CA 83 95            JZ   DECPT          ;/YES
9533    FE 15       TSTEX:  CPI  25Q            ;/TEST FOR E
9535    CA 98 95            JZ   INEXP          ;/YES - HANDLE EXP
9538    FE F0               CPI  360Q           ;/TEST FOR SPACE TERM (240B-260B)
953A    C2 1C 95            JNZ  ERR            ;/NOT LEGAL TERM
953D    CD 12 96            CALL FLTSGN         ;/FLOAT # AND SIGN IT
9540    CD BE 94    SCALE:  CALL GETEX          ;/GET DECIMAL EXP
9543    E6 7F               ANI  177Q           ;/GET GOOD BITS
9545    5F                  MOV  E,A            ;/SAVE COPY
9546    E6 40               ANI  100Q           ;/GET SIGN OF EXP
9548    07                  RLC                 ;/INTO SIGN BIT
9549    B7                  ORA  A              ;/SET FLOPS
954A    47                  MOV  B,A            ;/SAVE SIGN
954B    7B                  MOV  A,E            ;/GET EXP BACK
954C    CA 52 95            JZ   APLS           ;/JMP IS +
954F    3E 80               MVI  A,200Q         ;/MAKE MINUS +
9551    93                  SUB  E              ;/NOW ITS +
9552    80          APLS:   ADD  B              ;/SIGN NUMBER
9553    77                  MOV  M,A            ;/SAVE EXP (SIGN & MAG.)
9554    2E 14               MVI  L,(TEN5 AND 377Q)   ;/TRY MORD WITH 10**5 FIRST
9556    CD F1 94            CALL COPT           ;/TRANSFER TO RAM
9559    CD BE 94            CALL GETEX          ;/GET DECIMAL EXP
955C    E6 3F       INT5:   ANI  77Q            ;/GET MAG. OF EXP
955E    FE 05               CPI  5Q             ;/TEST FOR USE OF 10**5
9560    FA 6C 95            JM   TRYTN          ;/WONT GO - TRY 10
9563    CD A3 94            CALL MORD           ;/WILL GO SO DO IT
9566    D6 05               SUI  5Q             ;/MAG = MAG -5
9568    77                  MOV  M,A            ;/UPDATE DEC. EXP IN MEM
9569    C3 5C 95            JMP  INT5           ;/GO TRY AGAIN
956C    2E 18       TRYTN:  MVI  L,(TEN AND 377Q)   ;/PUT TEN IN RAM
956E    CD F1 94            CALL COPT
9571    CD BE 94            CALL GETEX          ;/SET UP FOR LOOP
9574    E6 3F       INT1:   ANI  77Q            ;/GET MAGNITUDE
9576    B7                  ORA  A              ;/TEST FOR 0
9577    CA 16 96            JZ   SAVEN          ;/DONE, MOVE NUM OUT AND GET OUT
957A    CD A3 94            CALL MORD           ;/NOT DONE - DO 10
957D    D6 01               SUI  1Q             ;/EXP = EXP -1
957F    77                  MOV  M,A            ;/UPDATE MEM
9580    C3 74 95            JMP  INT1           ;/TRY AGAIN
9583    69          DECPT:  MOV  L,C            ;/ZERO DIGIT COUNT
9584    2D                  DCR  L              ;/SINCE ITS NECESSARY
```

```
9585    2D                      DCR L           ;/TO COMPUTE EXP.
9586    36 00                   MVI M,0         ;/ZEROED
9588    CD 0C 96                CALL EP1        ;/GNUM IN MIDDLE
958B    5F                      MOV E,A         ;/SAVE TERMINATOR
958C    69                      MOV L,C         ;/MOVE DIGIT COUNT TO EXP
958D    2D                      DCR L           ;/BACK UP TO DIGIT COUNT
958E    2D                      DCR L
958F    46                      MOV B,M         ;/GOT DIGIT COUNT
9590    CD BE 94                CALL GETEX      ;/SET L TO DEC. EXP
9593    70                      MOV M,B         ;/PUT EXP
9594    7B                      MOV A,E         ;/TERM BACK TO A
9595    C3 33 95                JMP TSTEX       ;/TEST FOR E+OR-XX
9598    CD 12 96        INEXP:  CALL FLTSGN     ;/FLOAT AND SIGN NUMBER
959B    CD 16 96                CALL SAVEN      ;/SAVE NUMBER IN (L) TEMP
959E    CD 2D 96                CALL ZROIT      ;/ZERO OUT NUM. FOR INPUTTING EXP
95A1    CD C9 95                CALL GNUM       ;/NOW INPUT EXPONENT
95A4    FE F0                   CPI 360Q        ;/TEST FOR SPACE TERM.
95A6    C2 1C 95                JNZ ERR         ;/NOT LEGAL - TRY AGAIN
95A9    69                      MOV L,C         ;/GET EXP OUT OF MEM
95AA    2C                      INR L           ;/***TP
95AB    2C                      INR L           ;/EXP LIMITED TO 5 BITS
95AC    7E                      MOV A,M         ;/GET LOWEST 8 BITS
95AD    E6 1F                   ANI 37Q         ;/GET GOOD BITS
95AF    47                      MOV B,A         ;/SAVE THEM
95B0    2C                      INR L           ;/GET SIGN OF EXP
95B1    7E                      MOV A,M         ;/INTO A
95B2    B7                      ORA A           ;/SET FLOPS
95B3    78                      MOV A,B         ;/INCASE NOTHING TO DO
95B4    FA BA 95                JM  USEIT       ;/IF NEG. USE AS +
95B7    3E 00                   MVI A,0Q        ;/IF + MAKE -
95B9    90                      SUB B           ;/0-X =. -X
95BA    2C              USEIT:  INR L           ;/POINT AT EXP
95BB    86                      ADD M           ;/GET REAL DEC. EXP
95BC    77                      MOV M,A         ;/PUT IN MEM
95BD    79                      MOV A,C         ;/NOW GET NUMBER BACK
95BE    C6 0D                   ADI 15Q         ;/GET ADD OF L
95C0    6F                      MOV L,A         ;/L POINTS TO L OF NUMBER
95C1    6E                      MOV L,M         ;/NOW L POINTS TO NUMBER
95C2    7C                      MOV A,H         ;/RAM TO RAM COPY
95C3    CD FF 94                CALL COPY       ;/COPY IT BACK
95C6    C3 40 95                JMP SCALE       ;/NOW ADJUST FOR EXP
```

```
95C9  CD FA 97        GNUM:   CALL INP          ;/GET A CHAR
95CC  FE A0                   CPI 240Q          ;/IGNORE LEADING SPACES
95CE  CA C9 95                JZ  GNUM
95D1  FE AD                   CPI 255Q          ;/TEST FOR -
95D3  C2 DF 95                JNZ TRYP          ;/NOT MINUS
95D6  69                      MOV L,C           ;/MINUS SO SET SIGN
95D7  2C                      INR L             ;/IN CHAR LOC.
95D8  2C                      INR L             ;/***TP
95D9  2C                      INR L
95DA  36 80                   MVI M,200Q        ;/SET - SIGN
95DC  C3 C9 95                JMP GNUM
95DF  FE AB           TRYP:   CPI 253Q          ;/IGNORE +
95E1  CA C9 95                JZ  GNUM
95E4  D6 B0           TSTN:   SUI 260Q          ;/STRIP ASCII
95E6  F8                      RM                ;/RETURN IF TERM
95E7  FE 0A                   CPI 12Q           ;/TEST FOR NUMBER
95E9  F0                      RP                ;/ILLEGAL
95EA  5F                      MOV E,A           ;/SAVE DIGIT
95EB  CD 27 96                CALL GETN         ;/LOC. OF DIGIT STORAGE TO L
95EE  73                      MOV M,E           ;/SAVE DIGIT
95EF  CD 4E 94                CALL MULTT        ;/MULT NUMBER BY 10
95F2  B7                      ORA A             ;/TEST FOR TOO MANY DIGITS
95F3  C0                      RNZ               ;/TOO MANY DIGITS
95F4  CD 27 96                CALL GETN         ;/GET DIGIT
95F7  69                      MOV L,C           ;/SET L TO NUMBER
95F8  2C                      INR L
95F9  2C                      INR L             ;/***TP
95FA  86                      ADD M             ;/ADD IN THE DIGIT
95FB  77                      MOV M,A           ;/PUT RESULT BACK
95FC  2D                      DCR L             ;/NOW DO HIGH
95FD  7E                      MOV A,M           ;/GET HIGH TO ADD IN CARRY
95FE  CE 00                   ACI 0Q            ;/ADD IN CARRY
9600  77                      MOV M,A           ;/UPDATE HIGH
9601  2D                      DCR L             ;/***TP EXTENSION
9602  7E                      MOV A,M
9603  CE 00                   ACI 0Q            ;/ADD IN CARRY
9605  77                      MOV M,A           ;/***TP ALL DONE
9606  D8                      RC                ;/OVERFLOW ERROR
9607  2D                      DCR L             ;/BUMP DIGIT COUNT NOW
9608  2D                      DCR L
9609  46                      MOV B,M           ;/GET DIGIT COUNT
```

```
960A  04                      INR B             ;/BUMP DIGIT COUNT
960B  70                      MOV M,B           ;/UPDATE DIGIT COUNT
960C  CD FA 97       EP1:     CALL INP          ;/GET NEXT CHAR
960F  C3 E4 95                JMP TSTN          ;/MUST BE NUM. OR TERM
9612  69             FLTSGN:  MOV     L,C          ;POINT L AT NUMBER TO FLOAT
9613  C3 D5 91                JMP     FLOAT        ;GO FLOAT IT
9616  79              SAVEN:  MOV A,C           ;/PUT NUMBER IN (L)
9617  C6 0D                   ADI 15Q           ;/GET ADD OF L
9619  6F                      MOV L,A
961A  5E                      MOV E,M           ;/GET L OF RESULT
961B  6B                      MOV L,E           ;/POINT L AT (L)
961C  2C                      INR L             ;/SET TO 2ND WORD TO SAVE C
961D  71                      MOV M,C           ;/SAVE C IN (L)+1 SINCE IT WILL BE DESTROYED
961E  69                      MOV L,C           ;/SET UP TO CALL COPY
961F  4B                      MOV C,E           ;/NOW L&C SET
9620  7C                      MOV A,H           ;/RAM TO RAM COPY
9621  CD FF 94              CALL COPY          ;/COPY TO L
9624  4F                      MOV C,A           ;/(L)+1 RETURNED HERE SO SET AS C
9625  B7                       ORA     A              ;MAKE SURE CY=0 (NO ERROR)
9626  C9                      RET               ;/NOW EVERYTHING HUNKY-DORRY
9627  79              GETN:   MOV A,C           ;/GET DIGIT
9628  C6 0E                   ADI 16Q           ;/LAST LOC. IN SCRATCH
962A  6F                      MOV L,A           ;/PUT IN L
962B  7E                      MOV A,M           ;/GET DIGIT
962C  C9                      RET
962D  69              ZROIT:  MOV L,C           ;/ZERO NUMBER
962E  AF                      XRA A
962F  77                      MOV M,A           ;/***TP
9630  2C                      INR L             ;/***TP
9631  77                      MOV M,A
9632  2C                      INR L
9633  77                      MOV M,A
9634  2C                      INR L             ;/NOW SET SIGN TO +
9635  77                      MOV M,A
9636  C9                      RET               ;/DONE
                    ; CONTAIN LOW BYTE OF TWO BYTE VALUE. RETURNS CY=1 IF
                    ; BC>DE, CY=0 IF BC<DE: Z=1 IF BC=DE.
9637  7B             DCOMP:    MOV     A,E
9638  B9                       CMP     C
9639  C0                       RNZ
963A  7A                       MOV     A,D
```

```
963B  B8                        CMP     B
963C  C9                        RET
                        ; ROUTINE TO INPUT CHAR FROM TTY
963D  C5                CHAR2:  PUSH    B
963E  CD 04 01                  CALL    CONIN           ;INPUT FROM ODT
9641  78                        MOV     A,B             ;GET CHAR TO A REG.
9642  C1                        POP     B               ;RESTORE B,C
9643  C9                        RET
                        ; ROUTINE TO ADJUST VALUES OF BIN, FORWARD PNT. AND
                        ; LINE LENGTH OF SOURCE LINE.  PASSED ADD OF TEMP VARIABLE
                        ; CONTAINING ADD OF SOURCE LINE.
9644  F5                PTVAL:  PUSH    PSW
9645  D5                        PUSH    D
9646  E5                        PUSH    H
9647  3E 02                     MVI     A,002
9649  5E                        MOV     E,M
964A  2C                        INR     L
964B  56                        MOV     D,M
964C  2C                        INR     L
964D  D5                        PUSH    D
964E  E3                N1:     XTHL
964F  5E                        MOV     E,M
9650  23                        INX     H
9651  56                        MOV     D,M
9652  23                        INX     H
9653  E3                        XTHL
9654  73                        MOV     M,E
9655  2C                        INR     L
9656  72                        MOV     M,D
9657  2C                        INR     L
9658  3D                        DCR     A
9659  C2 4E 96                  JNZ     N1
965C  E3                        XTHL
965D  56                        MOV     D,M
965E  E1                        POP     H
965F  72                        MOV     M,D
9660  E1                        POP     H
9661  D1                        POP     D
9662  F1                        POP     PSW
9663  C9                        RET
                        ; ROUTINE TO CHK FLAGS ON INPUT AND OUTPUT.
```

```
                        ; PASSED FLAG VALUE IN REG B.
9664    F5              MCHK:   PUSH    PSW
9665    CD 0A 01        MCHK1:  CALL    STATUS
9668    A0                      ANA     B
9669    CA 65 96                JZ      MCHK1
966C    F1                      POP     PSW
966D    C9                      RET
                        ; MULTIPLICATION ROUTINE (ADD. VALUES)
966E    5E              MULT:   MOV     E,M
966F    2B                      DCX     H
9670    56                      MOV     D,M
9671    36 11                   MVI     M,11H
9673    06 00                   MVI     B,0
9675    48                      MOV     C,B
9676    7B              TOP:    MOV     A,E
9677    1F                      RAR
9678    5F                      MOV     E,A
9679    7A                      MOV     A,D
967A    1F                      RAR
967B    35                      DCR     M
967C    57                      MOV     D,A
967D    C8                      RZ
967E    D2 8B 96                JNC     SHIFT
9681    2B                      DCX     H
9682    2B                      DCX     H
9683    78                      MOV     A,B
9684    86                      ADD     M
9685    47                      MOV     B,A
9686    23                      INX     H
9687    79                      MOV     A,C
9688    8E                      ADC     M
9689    4F                      MOV     C,A
968A    23                      INX     H
968B    79              SHIFT:  MOV     A,C
968C    1F                      RAR
968D    4F                      MOV     C,A
968E    78                      MOV     A,B
968F    1F                      RAR
9690    47                      MOV     B,A
9691    C3 76 96                JMP     TOP
                        ;LINKAGES TO FLOATING POINT ROUTINES
```

```
97C7                                    ORG     113707Q
97C7    C3 AD 91                        JMP     NORM
97CA    C3 D5 91                        JMP     FLOAT
97CD    C3 8D 92                        JMP     WZER
97D0    C3 5C 90                        JMP     LADD
97D3    C3 5F 91                        JMP     LMUL
97D6    C3 00 90                        JMP     LDIV
97D9    C3 60 90                        JMP     LSUB
97DC    C3 CE 91                        JMP     DFXL
97DF    C3 57 91                        JMP     LMCM
97E2    C3 FF 94                        JMP     COPY
97E5    C3 9A 93                        JMP     CVRT
97E8    C3 1E 95                        JMP     INPUT
97EB    C3 6E 96                        JMP     MULT
97EE    C3 44 96                        JMP     PTVAL
97F1    C3 37 96                        JMP     DCOMP
97F4    C3 64 96                        JMP     MCHK
97F7    C3 3D 96                        JMP     CHAR2
97FA    C3 93 89                        JMP     INL
97FD    C3 D6 87                        JMP     OUTL
                                        END
NO PROGRAM ERRORS

                        SYMBOL TABLE

* 01

A       0007    ABCH    9134    ACPR    92EF    ADD1    94DC
ADD2    90DF    ADDZ    90D6    AORS    92E1    APLS    9552
B       0000    BBCH    913D    BCHK    91E2    BCTL    938D
BMIN    9122    C       0001    CCHK    9344    CCMP    925B
CFCHE   924C    CHAR2   963D    COM1    9116    COM2    9128
CONIN   0104    COPT    94F1    COPY    94FF    CRIN    904D
CSIGN   9369    CSTR    9371    CTWO    94E0    CVRT    939A
D       0002    DADD    9206    DCLR    921D    DCMP    92F4
DCOMP   9637    DECPT   9583    DECR    92D4 *  DFXL    91CE
DIGO    943A    DIVIT   94C3    DLST    91E9    DRST    91F8
DSUB    9226    DTST2   9012    E       0003    ENT1    9310
ENT2    930A    EP1     960C    EQ02    909C    EQUL    9094 *
ERR     951C    FINIT   94D7    FINUP   94B7    FLOAT   91D5
```

```
FLTSG  9612  FXL1   91B2  *      FXL2   91B3  GCHAR  9241
GCHR   949D  GETA   949E  GETEX  94BE  GETN   9627
GNUM   95C9  GØØN   903B  GØTV   93E5  H      0004  *
INCR   92B6  INCR2  92C6  INCR3  92C3  INDF1  9284
INDFC  92A8  INEXP  9598  INL    8993  INP    97FA
INPØP  9422  INPUT  951E  INT1   9574  INT5   955C
INTR   917F  KPGØ   9176  L003   0005  L008   9146
L001   9148  L002   911F  L003   914B  L128   914E
L129   9151  L131   9154  LADD   905C  LADS   9062
LASD   9102  LDCP   9339  LDIV   9000  LLTB   907F
LMCM   9157  LMCP   9340  LMUL   915F  LØØP   94E2
LSFT   9476  LSUB   9060  LXFR   9324  M      0006
MADD   91A4  MANT   90EF  MAXCH  903F  MCHK   9664
MCHK1  9665  MDGN   9400  MDSKP  940E  MINCH  00C0
MØRD   94A3  MSFH   9379  MULT   966E  MULTT  944E
N1     964E  NCHK   9084  NNZRØ  93AF  NØRM   91AD
NØRM1  91AE  NØTO   90AA  ØFLW1  9276  ØK1    9409
ØUTL   87D6  ØUTR   97FD  ØVER   9317  PLSV   9499
PØPD   941F  PSW    0006  PTVAL  9644  REP3   9027
REP5   9326  REP6   91B9  SAVEN  9616  SCALE  9540
SCCFG  92D2  SCHAR  91CB  SHIFT  968B  SHIØ   908C
SIGN   9490  SP     0006  STØRC  9353  STATU  010A
SUBZ   90B9  TEN    9518  TLØØP  9479  TLP2   947A
TØP    9676  TRY1   93F3  TRYP   95DF  TRYTN  956C
TST1   93F8  TST8   93D8  TSTEX  9533  TSTR   92E7
TWØD   94C9  UFLW1  9268  UFLWC  9362  USEIT  95BA
WCHAR  929F  WFLT   0B57  WIND   927F  WMANT  9298
WØVR   9271  WUND   9263  WZER   928D  WZERC  92AF
ZCHK   91DA  ZMCHK  91DA  ZRØIT  962D

*  02
*  03
*  04
```

Chapter 2

Dr. Wang's Palo Alto Tiny Basic

By Roger Rauskolb

Tiny Basic was first proposed in Dr. Dobb's Journal. Li-Chen Wang's version of Palo Alto Tiny Basic originally appeared in Issue No. 5, May 1976 of Dr. Dobb's Journal. A complete listing was printed, but Dr. Wang did the assembly on an IBM computer and he defined different mnemonics. In order to assemble with an Intel Compatible Assembler a translation of most mnemonics had to be performed.

I had developed my own system, which consists of two small p.c. boards, one containing the 8080 CPU, 4k of 2708 type EROM and 1K of RAM. The other PCB contained the RS-232 Interface using the Intel 8251. So I wanted to change the I/O section.

If you want to change I/O, all routines are contained in the OUTC and CHKIO routines. My system uses the following configuration:

2708	EROMS	0000H	TO	07FFH
1k	OF RAM	1000H	TO	13FFH
8251	DATA PORT	0FAH		
8251	STATUS PORT			

Command Instruction 27H = 2 stop bits parity disabled.
8 bit characters.
Baud Rate Factor of 04.

Mode Instruction OCFH = No Hunt Mode. Not (RTS) forced to 0. Receive Enabled Data Terminal Ready Transmit Enabled.

Transmitter Ready Status Bit = Bit 0 (01H)
Receiver Buffer Ready Status Bit = Bit 1 (02H)

The program is contained in locations 0000H to 0768H.

In 1K of RAM 847 bytes are left over for program.

Tiny Basic does not offer much in terms of functions and general mathematical capabilities. But it is great to teach programming basic

to children (and adults) and for games, since it has the RND function. It takes up little memory space and executes a lot faster than other basics.

Dr. Wang was very helpful and assisted me all the way. Some errors were eliminated. I appreciate his help and he deserves a lot of credit for his implementation of Tiny Basic.

THE TINY BASIC LANGUAGE

Numbers

In Tiny Basic, all numbers are integers and must be less than or equal to 32767.

Variables

There are 26 variables denoted by letters A through Z. There is also a single array @(1). The dimension of this array (i.e., the range of value of the index 1) is set automatically to make use of all the memory space that is left unused by the program. (i.e., 0 through SIZE/2, see SIZE function below.)

Functions

For the time being, there are only 3 functions:

ABS(X) gives the absolute value of X.
RND(X) gives a random number between 1 and X (inclusive).
SIZE gives the number of bytes left unused by the program.

Arithmetic and Compare Operators

/ divide. Note that since we have integers only, 2/3 = 0.
* multiply.
– subtract.
+ add.
> compare if greater than.
< compare if less than.
= compare if equal to. Note than to certain versions of Basic "LET A = B = O" means "set both A and B to O." To this version of Tiny Basic, it means "set A to the result of comparing B with O."
compare if not equal to.
>= compare if greater than or equal to.
<= compare if less than or equal to.

+, –, *, and / operations result in a value of between –32767 and 32767. All compare operators result in a 1 if true and a 0 if not true.

Expressions

Expressions are formed with numbers, variables, and functions with arithmetic and compare operators between them. + and – signs

can also be used at the beginning of an expression. The value of an expression is evaluated from left to right, except that * and / are always done first, and then + and −, and then compare operators. Parentheses can also be used to alter the order of evaluation.

Statements

A Tiny Basic statement consists of a statement number between 1 and 32767 followed by one or more commands. Commands in the same statement are separated by a semi-colon ";". "GOTO", "STOP", and "RETURN" commands must be the last command in any given statement.

Program

A Tiny Basic program consists of one or more statements. When a direct command "RUN" is issued, the statement with the lowest statement number is executed first, then the one with the next lowest statement number, etc. However, the "GOTO", "GOSUB", "STOP", and "RETURN" commands can alter this normal sequence. Within the statement, execution of the commands is from left to right. The "IF" command can cause the execution of all the commands to its right in the same statement to be skipped over.

Commands

Tiny Basic commands are listed below with examples. Remember that commands can be concatenated with semi-colons. In order to store the statement, you must also have a statement number in front of the commands. The statement number and the concatenation are not shown in the examples.

REM or REMARK Command

```
REM anything goes
This line will be ignored by TBI.
```

LET Command

```
LET A=234-5*6, A=A/2, X=A-100, @(X+9)=A-1
```

will set the variable A to the value of the expression 234 − 5*6 (i.e., 204), set the variable A (again) to the value of the expression A/2 (i.e., 102), set the variable X to the value of the expression A-100 (i.e., 2), and then set the variable @(11) to 101 (where 11 is the value of the expression X + 9 and 101 is the value of the expression A − 1).

```
LET U=A#B, V=(A>B)*X+(A<B)*Y
```

will set the variable U to either 1 or 0 depending on whether A is not equal to or is equal to B; and set the variable V to either X, Y or 0 depending on whether A is greater than, less than, or equal to B.

PRINT Command

PRINT

will cause a carriage-return (CR) and a line-feed (LF) on the output device.

PRINT A*3+1,"ABC 123 !@#",'CBA'

will print the value of the expression A*3+1 (i.e., 307), the string of characters "ABC 123 !@#", and the string "CBA", and then a CR-LF. Note that either single or double quotes can be used to quote strings, but pairs must be matched.

PRINT A*3+1,"ABC 123 !@#",'CBA',

will produce the same output as before, except that there is no CR-LF after the last item is printed. This enables the program to continue printing on the same line with another "PRINT".

PRINT A, B, #3, C, D, E, #10, F, G

will print the values of A and B in 6 spaces, the values of C, D, and E in 3 spaces, and the values of F and G in 10 spaces. If there are not enough spaces specified for a given value to be printed, the value will be printed with enough spaces anyway.

PRINT 'ABC',←,'XXX'

will print the string "ABC", a CR without a LF, and then the string "XXX" (over the ABC) followed by a CR-LF.

INPUT Command

INPUT A,B

When this command is executed, Tiny Basic will print "A:" and wait to read in an expression from the input device. The variable A will be set to the value of this expression. Then "B:" is printed and variable B is set to the value of the next expression read from the input device. Note that not only numbers, but also expressions can be read as input.

INPUT 'WHAT IS THE WEIGHT'A, "AND SIZE"B

This is the same as the command above, except the prompt "A:" is replaced by "WHAT IS THE WEIGHT:" and the prompt "B:" is replaced by "AND SIZE:". Again, both single and double quotes can be used as long as they are matched.

INPUT A, 'STRING',←,"ANOTHER STRING",B

the strings and the "←" have the same effect as in "PRINT".

IF Command

IF A<B LET X=3; PRINT 'THIS STRING'

will test the value of the expression A<B. If it is not zero (i.e., if it is true), the commands in the rest of this statement will be executed. If the value of the expression is zero (i.e., if it is not true), the rest of this statement will be skipped over and execution continues at next statement. Note that the word "THEN" is not used.

GOTO Command

GOTO 120

will cause the execution to jump to statement 120. Note that GOTO command cannot be followed by a semi-colon and other commands. It must be ended with a CR.

GOTO A*10+B

will cause the execution to jump to a different statement number as computed from the value of the expression.

GOSUB and RETURN Commands

GOSUB command is similar to GOTO command except that: a) the current statement number and position within the statement is remembered; and b) a semi-colon and other commands can follow it in the same statement.

GOSUB 120

will cause the execution to jump to statement 120.

GOSUB A*10+B

will cause the execution to jump to different statements as computed from the value of the expression a*10+B.

RETURN

A RETURN command must be the last command in a statement and followed by a CR. When a RETURN command is encountered, it will cause the execution to jump back to the command following the most recent GOSUB command.

GOSUB can be nested. The depth of nesting is limited only by the stack space.

LIST

will print out all the statements in numerical order.

LIST 120

will print out all the statements in numerical order 120.

NEW

will delete all the statements.

Stopping the Execution

The execution of program or listing of program can be stopped by the Control-C key on the input device.

Abbreviation and blanks

You may use blanks freely, except that numbers, command key words, and function names can not have embedded blanks.

You can truncate all command key words and function names and follow each by a period. "P.", "PR.", "PRI.", and "PRIN." all stand for "PRINT." Also the word LET in LET command can be omitted. The "shortest" abbreviation for all the key words are as follows:

A. = ABS	F. = FOR	GOS. = GOSUB	G. = GOTO
IF = IF	IN. = INPUT	L. = LIST	N. = NEW
N. = NEXT	P. = PRINT	REM. = REMARK	R. = RETURN
R. = RND	R. = RUN	S. = SIZE	S. = STEP
S. = STOP	TO = TO		
Implied = LET			

Control of Output Device

The Control-O key on the input device can be used to turn the output device ON and OFF. This is useful when you want to read in a program punched on paper tape.

To produce such a paper tape, type "LIST" without CR. Turn on the paper tape punch and type a few Control-Shift-P's and then a CR. When listing is finished, type more Control-Shift-P's and turn off the punch.

To read back such a paper tape, type "NEW," CR, and Control-O, then turn on the paper tape reader. When the paper tape is finished, turn it off and type a Control-O again.

Control-Shift-P's and turn off the punch

To read back such a paper tape, type "NEW," CR, and Control-?, then turn on the paper tape reader. When the paper tape is finished, turn it off and type a Control-O, then turn on the paper tape reader. When the paper tape is finished, turn it off and type a Control-O again.

Error Report

There are only three error conditions in TINY BASIC. The statement with the error is printed out with a question mark inserted at the point where the error is detected.

(1) WHAT? means it does not understand you. Example:

WHAT? 210 P?TINT "THIS" where PRINT is mistyped

WHAT? 260 LET A = B + 3, C = (3 + 4), X = 4

(2) HOW? means it understands you but does not know how to do it.

HOW? 310LET A = B*C? + 2 where B*C is larger than 32767

HOW? 380 GOTO 412? where 412 does not exist

(3) SORRY means it understands you and knows how to do it but there is not enough memory to do it.

Error Corrections

If you notice an error in typing before you hit the CR, you can delete the last character by the Rub-Out key or delete the entire line by the Alt-Mode key. Tiny basic will echo a back-slash for each Rub-Out. Echo for Alt-Mode consists of a LF, a CR, and an up-arrow.

To correct a statement, you can retype the statement number and the correct commands. Tiny Basic will replace the old statement with the new one.

To delete a statement, type the statement number and a CR only.

Verify the corrections by "LIST nnnn" and hit the Control-C key while the line is being printed.

FOR and NEXT Commands

FOR X = A + 1 TO 3*B STEP C-1

The variable X is set to the value of the expression A + 1. The values of the expressions (not the expressions themselves) 3*B and C-1 are remembered. The name of the variable X, the statement number and the position of this command within the statement are also remembered. Execution then continues the normal way until a NEXT command is encountered.

The STEP can be positive, negative or even zero. The word STEP and the expression following it can be omitted if the desired STEP is + 1.

NEXT X

The name of the variable (X) is checked with that of the most recent FOR command. If they do not agree, that FOR is terminated and the next recent FOR is checked, etc. When a match is found, this variable will be set to its current value plus the value of the STEP expression saved by the FOR command. The updated value is then compared with the value of the TO expression also saved by the FOR command. If this is within the limit, execution will jump back to the command following the FOR command. If this is outside the limit, execution continues following the NEXT command itself.

FOR can be nested. The depth of nesting is limited only by the stack space. If a new FOR command with the same control variable as that of an old FOR command is encountered, the old FOR will be terminated automatically.

STOP Command

STOP

This command stops the execution of the program and returns control to direct commands from the input device. It can appear many times in a program but must be the last command in any given statement, i.e., it cannot be followed by semi-colon and other commands.

Direct Commands

As defined earlier, a statement consists of a statement number followed by commands. IF the statement number is missing, or if it is 0, the commands will be executed after you have typed the CR. All the commands described above can be used as direct commands. There are three more commands that can be used as direct command but not as part of a statement:

RUN

will start to execute the program starting at the lowest statement number.

```
CP/M MACRO ASSEM 2.0    #001    TINY BASIC FOR INTEL 8080

                ;*****************************************************************
                ;*
                        TITLE   'TINY BASIC FOR INTEL 8080'
                ;*                      VERSION 1.0
                ;*                      BY LI-CHEN WANG
                ;*                      10 JUNE, 1976
                ;*                      @COPYLEFT
                ;*                      ALL WRONGS RESERVED
                ;*
                ;*****************************************************************
                ;*
                ;*      ;*** ZERO PAGE SUBROUTINES ***
                ;*
                ;*  THE 8080 INSTRUCTION SET LETS YOU HAVE 8 ROUTINES IN LOW
                ;*  MEMORY THAT MAY BE CALLED BY RST N, N BEING 0 THROUGH 7.
                ;*  THIS IS A ONE-BYTE INSTRUCTION AND HAS THE SAME POWER AS
                ;*  THE THREE-BYTE INSTRUCTION CALL  LLHH.  TINY BASIC WILL
                ;*  USE RST 0 AS START AND RST 1 THROUGH RST 7 FOR
                ;*  THE SEVEN MOST FREQUENTLY USED SUBROUTINES.
                ;*  TWO OTHER SUBROUTINES (CRLF AND TSTNUM) ARE ALSO IN THIS
                ;*  SECTION.  THEY CAN BE REACHED ONLY BY 3-BYTE CALLS.
                ;*  IN ORDER TO CONFIGURE THE SYSTEM FOR USE WITH CPM I HAVE
                ;*  MOVED SOME OF THE ROUTINES AROUND.  START WILL NOW BE AT
                ;*  LOCATION 100H AND THIS SECTION WILL END AT LOCATION 3FH
                ;*  WITH A JUMP TO 108H.
                ;*
                ;       ORG     8H
                ;       XTHL            ;*** TSTC OR RST 1 ***
                ;       RST     5       ;IGNORE BLANKS AND
                ;       CMP     M       ;TEST CHARACTER
                ;       JMP     TC1     ;REST OF THIS IS AT TC1
                ;*
                ;CRLF   MVI     A,0DH   ;*** CRLF ***
                ;*
                ;       PUSH    PSW     ;*** OUTC OR RST 2 ***
                ;       LDA     OCSW    ;PRINT CHARACTER ONLY
                ;       ORA     A       ;IFF OCSW SWITCH IS ON
                ;       JMP     OC2     ;REST OF THIS IS AT OC2
                ;*
                ;       CALL    EXPR2   ;*** EXPR OR RST 3 ***
                ;       PUSH    H       ;EVALUATE AN EXPRESION
                ;       JMP     EXPR1   ;REST OF IT IS AT EXPR1
                ;       DB      'W'
                ;*
                ;       MOV     A,H     ;*** COMP OR RST 4 ***
                ;       CMP     D       ;COMPARE HL WITH DE
                ;       RNZ             ;RETURN CORRECT C AND
                ;       MOV     A,L     ;Z FLAGS
                ;       CMP     E       ;BUT OLD A IS LOST
                ;       RET
                ;       DB      'AN'
                ;*
                ;SS1    LDAX    D       ;*** IGNBLK/RST 5 ***
                ;       CPI     40Q     ;IGNORE BLANKS
                ;       RNZ             ;IN TEXT (WHERE DE->)
                ;       INX     D       ;AND RETURN THE FIRST
                ;       JMP     SS1     ;NON-BLANK CHAR. IN A
                ;*
                ;       POP     PSW     ;*** FINISH/RST 6 ***
                ;       CALL    FIN     ;CHECK END OF COMMAND
                ;       JMP     QWHAT   ;PRINT "WHAT?" IFF WRONG
                ;       DB      'G'
                ;*
                ;       RST     5       ;*** TSTV OR RST 7 ***
                ;       SUI     100Q    ;TEST VARIABLES
                ;       RC              ;C:NOT A VARIABLE
                ;       JMP     TSTV1   ;JUMP AROUND RESERVED AREA
0100                    ORG     100H    ;OF CPM.
0100 C3A00A     START   JMP     NINIT   ;GO TO INITIALIZATION ROUTINE.  JIF
0103 C21F01     TSTV1   JNZ     TV1     ;NOT '@' ARRAY
0106 13                 INX     D       ;IT IS THE '@' ARRAY
0107 CDBC06             CALL    PARN    ;@ SHOULD BE FOLLOWED
010A 29                 DAD     H       ;BY (EXPR) AS ITS INDEX
010B DA6601             JC      QHOW    ;IS INDEX TOO BIG?
010E D5                 PUSH    D       ;WILL IT OVERWRITE
010F EB                 XCHG            ;TEXT?
0110 CDFE06             CALL    SIZE    ;FIND SIZE OF FREE
0113 E7                 RST     4       ;AND CHECK THAT
0114 DA0E08             JC      ASORRY  ;IFF SO, SAY "SORRY"
0117 21000F     SS1A    LXI     H,VARBGN ;IFF NOT, GET ADDRESS
```

```
011A CD9C07             CALL    SUBDE   ;OF @(EXPR) AND PUT IT
011D D1                 POP     D       ;IN HL
011E C9                 RET             ;C FLAG IS CLEARED
011F FE1B       TV1     CPI     33Q     ;NOT @, IS IT A TO Z?
0121 3F                 CMC             ;IFF NOT RETURN C FLAG
0122 D8                 RC
0123 13                 INX     D       ;IFF A THROUGH Z
0124 21000F     TV1A    LXI     H,VARBGN ;COMPUTE ADDRESS OF
0127 07                 RLC             ;THAT VARIABLE
0128 85                 ADD     L       ;AND RETURN IT IN HL
0129 6F                 MOV     L,A     ;WITH C FLAG CLEARED
012A 3E00               MVI     A,0
012C 8C                 ADC     H
012D 67                 MOV     H,A
012E C9                 RET
                ;*
                ;*       TSTC    XCH     HL,(SP) ;*** TSTC OR RST 1 ***
                ;*               IGNBLK  THIS IS AT LOC. 8
                ;*               CMP     M       AND THEN JMP HERE
012F 23         TC1     INX     H       ;COMPARE THE BYTE THAT
0130 CA3A01             JZ      TC2     ;FOLLOWS THE RST INST.
0133 C5                 PUSH    B       ;WITH THE TEXT (DE->)
0134 4E                 MOV     C,M     ;IFF NOT =, ADD THE 2ND
0135 0600               MVI     B,0     ;BYTE THAT FOLLOWS THE
0137 09                 DAD     B       ;RST TO THE OLD PC
0138 C1                 POP     B       ;I.E., DO A RELATIVE
0139 1B                 DCX     D       ;JUMP IFF NOT =
013A 13         TC2     INX     D       ;IFF =, SKIP THOSE BYTES
013B 23                 INX     H       ;AND CONTINUE
013C E3                 XTHL
013D C9                 RET
                ;*
013E 210000     TSTNUM  LXI     H,0     ;*** TSTNUM ***
0141 44                 MOV     B,H     ;TEST IFF THE TEXT IS
0142 EF                 RST     5       ;A NUMBER
0143 FE30       TN1     CPI     60Q     ;IFF NOT, RETURN 0 IN
0145 D8                 RC              ;B AND HL
0146 FE3A               CPI     72Q     ;IFF NUMBERS, CONVERT
0148 D0                 RNC             ;TO BINARY IN HL AND
0149 3EF0               MVI     A,360Q  ;SET A TO # OF DIGITS
014B A4                 ANA     H       ;IFF H>255, THERE IS NO
014C C26601             JNZ     QHOW    ;ROOM FOR NEXT DIGIT
014F 04                 INR     B       ;B COUNTS # OF DIGITS
0150 C5                 PUSH    B
0151 44                 MOV     B,H     ;HL=10;*HL+(NEW DIGIT)
0152 4D                 MOV     C,L
0153 29                 DAD     H       ;WHERE 10;* IS DONE BY
0154 29                 DAD     H       ;SHIFT AND ADD
0155 09                 DAD     B
0156 29                 DAD     H
0157 1A                 LDAX    D       ;AND (DIGIT) IS FROM
0158 13                 INX     D       ;STRIPPING THE ASCII
0159 E60F               ANI     17Q     ;CODE
015B 85                 ADD     L
015C 6F                 MOV     L,A
015D 3E00               MVI     A,0
015F 8C                 ADC     H
0160 67                 MOV     H,A
0161 C1                 POP     B
0162 1A                 LDAX    D       ;DO THIS DIGIT AFTER
0163 F24301             JP      TN1     ;DIGIT. S SAYS OVERFLOW
0166 D5         QHOW    PUSH    D       ;*** ERROR: "HOW?" ***
0167 116D01     AHOW    LXI     D,HOW
016A C3E407             JMP     ERROR
016D 484F573F0DHOW      DB      'HOW?',0DH
0172 4F4B0D     OK      DB      'OK',0DH
0175 574841543FWHAT     DB      'WHAT?',0DH
017B 534F525259SORRY    DB      'SORRY',0DH
                ;*
                ;*****************************************************************
                ;*
                ;* *** MAIN ***
                ;*
                ;* THIS IS THE MAIN LOOP THAT COLLECTS THE TINY BASIC PROGRAM
                ;* AND STORES IT IN THE MEMORY.
                ;*
                ;* AT START, IT PRINTS OUT "(CR)OK(CR)", AND INITIALIZES THE
                ;* STACK AND SOME OTHER INTERNAL VARIABLES. THEN IT PROMPTS
                ;* ">" AND READS A LINE. IFF THE LINE STARTS WITH A NON-ZERO
                ;* NUMBER, THIS NUMBER IS THE LINE NUMBER. THE LINE NUMBER
                ;* (IN 16 BIT BINARY) AND THE REST OF THE LINE (INCLUDING CR)
                ;* IS STORED IN THE MEMORY. IFF A LINE WITH THE SAME LINE
```

```
                        ;* NUMBER IS ALREDY THERE, IT IS REPLACED BY THE NEW ONE. IF
                        ;* THE REST OF THE LINE CONSISTS OF A ODHONLY, IT IS NOT STORED
                        ;* AND ANY EXISTING LINE WITH THE SAME LINE NUMBER IS DELETED.
                        ;*
                        ;* AFTER A LINE IS INSERTED, REPLACED, OR DELETED, THE PROGRAM
                        ;* LOOPS BACK AND ASK FOR ANOTHER LINE. THIS LOOP WILL BE
                        ;* TERMINATED WHEN IT READS A LINE WITH ZERO OR NO LINE
                        ;* NUMBER; AND CONTROL IS TRANSFERED TO 'DIRCT'.
                        ;*
                        ;* TINY BASIC PROGRAM SAVE AREA STARTS AT THE MEMORY LOCATION
                        ;* LABELED 'TXTBGN' AND ENDED AT 'TXTEND'. WE ALWAYS FILL THIS
                        ;* AREA STARTING AT 'TXTBGN', THE UNFILLED PORTION IS POINTED
                        ;* BY THE CONTENT OF A MEMORY LOCATION LABELED 'TXTUNF'.
                        ;*
                        ;* THE MEMORY LOCATION 'CURRNT' POINTS TO THE LINE NUMBER
                        ;* THAT IS CURRENTLY BEING INTERPRETED. WHILE WE ARE IN
                        ;* THIS LOOP OR WHILE WE ARE INTERPRETING A DIRECT COMMAND
                        ;* (SEE NEXT SECTION), 'CURRNT' SHOULD POINT TO A 0.
                        ;*
0181 310020     RSTART  LXI     SP,STACK ;SET STACK POINTER
0184 CD0E00     ST1     CALL    CRLF     ;AND JUMP TO HERE
0187 117201             LXI     D,OK     ;DE->STRING
018A 97                 SUB     A        ;A=0
018B CD7F08             CALL    PRTSTG   ;PRINT STRING UNTIL ODH
018E 219501             LXI     H,ST2+1  ;LITERAL 0
0191 22C109             SHLD    CURRNT   ;CURRNT->LINE # = 0
0194 210000     ST2     LXI     H,0
0197 22C909             SHLD    LOPVAR
019A 22C309             SHLD    STKGOS
019D 3E3E       ST3     MVI     A,76Q    ;PROMPT '>' AND
019F CD1408             CALL    GETLN    ;READ A LINE
01A2 D5                 PUSH    D        ;DE->END OF LINE
01A3 11370F     ST3A    LXI     D,BUFFER ;DE->BEGINNING OF LINE
01A6 CD3E01             CALL    TSTNUM   ;TEST IFF IT IS A NUMBER
01A9 EF                 RST     5
01AA 7C                 MOV     A,H      ;HL=VALUE OF THE # OR
01AB B5                 ORA     L        ;0 IFF NO # WAS FOUND
01AC C1                 POP     B        ;BC->END OF LINE
01AD CAB502             JZ      DIRECT
01B0 1B                 DCX     D        ;BACKUP DE AND SAVE
01B1 7C                 MOV     A,H      ;VALUE OF LINE # THERE
01B2 12                 STAX    D
01B3 1B                 DCX     D
01B4 7D                 MOV     A,L
01B5 12                 STAX    D
01B6 C5                 PUSH    B        ;BC,DE->BEGIN, END
01B7 D5                 PUSH    D
01B8 79                 MOV     A,C
01B9 93                 SUB     E
01BA F5                 PUSH    PSW      ;A=# OF BYTES IN LINE
01BB CD5708             CALL    FNDLN    ;FIND THIS LINE IN SAVE
01BE D5                 PUSH    D        ;AREA, DE->SAVE AREA
01BF C2D201             JNZ     ST4      ;NZ:NOT FOUND, INSERT
01C2 D5                 PUSH    D        ;Z:FOUND, DELETE IT
01C3 CD7308             CALL    FNDNXT   ;FIND NEXT LINE
                ;*      DE->NEXT LINE
01C6 C1                 POP     B        ;BC->LINE TO BE DELETED
01C7 2AD509             LHLD    TXTUNF   ;HL->UNFILLED SAVE AREA
01CA CD0009             CALL    MVUP     ;MOVE UP TO DELETE
01CD 60                 MOV     H,B      ;TXTUNF->UNFILLED AREA
01CE 69                 MOV     L,C
01CF 22D509             SHLD    TXTUNF   ;UPDATE
01D2 C1         ST4     POP     B        ;GET READY TO INSERT
01D3 2AD509             LHLD    TXTUNF   ;BUT FIRT CHECK IF
01D6 F1                 POP     PSW      ;THE LENGTH OF NEW LINE
01D7 E5                 PUSH    H        ;IS 3 (LINE # AND CR)
01D8 FE03               CPI     3        ;THEN DO NOT INSERT
01DA CA8101             JZ      RSTART   ;MUST CLEAR THE STACK
01DD 85                 ADD     L        ;COMPUTE NEW TXTUNF
01DE 6F                 MOV     L,A
01DF 3E00               MVI     A,0
01E1 8C                 ADC     H
01E2 67                 MOV     H,A      ;HL->NEW UNFILLED AREA
01E3 11000F     ST4A    LXI     D,TXTEND ;CHECK TO SEE IF THERE
01E6 E7                 RST     4        ;IS ENOUGH SPACE
01E7 D20D08             JNC     QSORRY   ;SORRY, NO ROOM FOR IT
01EA 22D509             SHLD    TXTUNF   ;OK, UPDATE TXTUNF
01ED D1                 POP     D        ;DE->OLD UNFILLED AREA
01EE CD0909             CALL    MVDOWN
01F1 D1                 POP     D        ;DE->BEGIN, HL->END
01F2 E1                 POP     H
01F3 CD0009             CALL    MVUP     ;MOVE NEW LINE TO SAVE
01F6 C39D01             JMP     ST3      ;AREA
```

```
                ;*
                ;*****************************************************************
                ;*
                ;* *** TABLES *** DIRECT *** & EXEC ***
                ;*
                ;* THIS SECTION OF THE CODE TESTS A STRING AGAINST A TABLE.
                ;* WHEN A MATCH IS FOUND, CONTROL IS TRANSFERED TO THE SECTION
                ;* OF CODE ACCORDING TO THE TABLE.
                ;*
                ;* AT 'EXEC', DE SHOULD POINT TO THE STRING AD HL SHOULD POINT
                ;* TO THE TABLE-1. AT 'DIRECT', DE SHOULD POINT TO THE STRING,
                ;* HL WILL BE SET UP TO POINT TO TAB1-1, WHICH IS THE TABLE OF
                ;* ALL DIRECT AND STATEMENT COMMANDS.
                ;*
                ;* A '.' IN THE STRING WILL TERMINATE THE TEST AND THE PARTIAL
                ;* MATCH WILL BE CONSIDERED AS A MATCH. E.G., 'P.', 'PR.',
                ;* 'PRI.', 'PRIN.', OR 'PRINT' WILL ALL MATCH 'PRINT'.
                ;*
                ;* THE TABLE CONSISTS OF ANY NUMBER OF ITEMS. EACH ITEM
                ;* IS A STRING OF CHARACTERS WITH BIT 7 SET TO 0 AND
                ;* A JUMP ADDRESS STORED HI-LOW WITH BIT 7 OF THE HIGH
                ;* BYTE SET TO 1.
                ;*
                ;* END OF TABLE IS AN ITEM WITH A JUMP ADDRESS ONLY. IFF THE
                ;* STRING DOES NOT MATCH ANY OF THE OTHER ITEMS, IT WILL
                ;* MATCH THIS NULL ITEM AS DEFAULT.
                ;*
01F9 =          TAB1    EQU     $       ;DIRECT COMMANDS
01F9 4C495354           DB      'LIST'
01FD 8420               DB      LIST SHR 8 + 128,LIST AND 0FFH
01FF 52554E             DB      'RUN'
0202 8315               DB      RUN SHR 8 + 128,RUN AND 255
0204 4E4557             DB      'NEW'
0207 8306               DB      NEW SHR 8 + 128,NEW AND 255
0209 4C4F4144           DB      'LOAD'
020D 8343               DB      DLOAD SHR 8 + 128,DLOAD AND 255
020F 53415645           DB      'SAVE'
0213 838A               DB      DSAVE SHR 8 + 128,DSAVE AND 255
0215 4259458000         DB      'BYE',80H,0H    ;GO BACK TO CPM
021A =          TAB2    EQU     $       ;DIRECT/TATEMENT
021A 4E455854           DB      'NEXT'
021E 8508               DB      NEXT SHR 8 + 128,NEXT AND 255
0220 4C4554             DB      'LET'
0223 85C8               DB      LET SHR 8 + 128,LET AND 255
0225 4F5554             DB      'OUT'
0228 870B               DB      OUTCMD SHR 8 + 128,OUTCMD AND 255
022A 504F4B45           DB      'POKE'
022E 8747               DB      POKE SHR 8 + 128,POKE AND 255
0230 57414954           DB      'WAIT'
0234 871F               DB      WAITCM SHR 8 + 128,WAITCM AND 255
0236 4946               DB      'IF'
0238 8559               DB      IFF SHR 8 + 128,IFF AND 255
023A 474F544F           DB      'GOTO'
023E 8334               DB      GOTO SHR 8 + 128,GOTO AND 255
0240 474F535542         DB      'GOSUB'
0245 8470               DB      GOSUB SHR 8 + 128,GOSUB AND 255
0247 5245545552         DB      'RETURN'
024D 8490               DB      RETURN SHR 8 + 128,RETURN AND 255
024F 52454D             DB      'REM'
0252 8555               DB      REM SHR 8 + 128,REM AND 255
0254 464F52             DB      'FOR'
0257 84A9               DB      FOR SHR 8 + 128,FOR AND 255
0259 494E505554         DB      'INPUT'
025E 8572               DB      INPUT SHR 8 + 128,INPUT AND 255
0260 5052494E54         DB      'PRINT'
0265 8438               DB      PRINT SHR 8 + 128,PRINT AND 255
0267 53544F50           DB      'STOP'
026B 830F               DB      STOP SHR 8 + 128,STOP AND 255
026D 85C2               DB      DEFLT SHR 8 + 128,DEFLT AND 255
026F 594F552043         DB      'YOU CAN ADD MORE';COMMANDS BUT
                                ;REMEMBER TO MOVE DEFAULT DOWN.
027F =          TAB4    EQU     $       ;FUNCTIONS
027F 524E44             DB      'RND'
0282 86C7               DB      RND SHR 8 + 128,RND AND 255
0284 494E50             DB      'INP'
0287 8738               DB      INP SHR 8 + 128,INP AND 255
0289 5045454B           DB      'PEEK'
028D 8757               DB      PEEK SHR 8 + 128,PEEK AND 255
028F 555352             DB      'USR'
0292 8761               DB      USR SHR 8 + 128,USR AND 255
0294 414253             DB      'ABS'
0297 86F2               DB      ABS SHR 8 + 128,ABS AND 255
```

```
0299 53495A45          DB      'SIZE'
029D 86FE              DB      SIZE SHR 8 + 128,SIZE AND 255
029F 86AD              DB      XP40 SHR 8 + 128,XP40 AND 255
02A1 594F552043        DB      'YOU CAN ADD MORE';FUNCTIONS BUT REMEMBER
                                       ;TO MOVE XP40 DOWN
02B1 =          TAB5   EQU     $       ;"TO" IN "FOR"
02B1 544F              DB      'TO'
02B3 84B9              DB      FR1 SHR 8 + 128,FR1 AND 255
02B5 87E0              DB      QWHAT SHR 8 + 128,QWHAT AND 255
02B7 =          TAB6   EQU     $       ;"STEP" IN "FOR"
02B7 53544550          DB      'STEP'
02BB 84C3              DB      FR2 SHR 8 + 128,FR2 AND 255
02BD 84C7              DB      FR3 SHR 8 + 128,FR3 AND 255
02BF =          TAB8   EQU     $       ;RELATION OPERATORS
02BF 3E3D              DB      '>='
02C1 85D8              DB      XP11 SHR 8 + 128,XP11 AND 255
02C3 23                DB      '#'
02C4 85DE              DB      XP12 SHR 8 + 128,XP12 AND 255
02C6 3E                DB      '>'
02C7 85E4              DB      XP13 SHR 8 + 128,XP13 AND 255
02C9 3D                DB      '='
02CA 85F3              DB      XP15 SHR 8 + 128,XP15 AND 255
02CC 3C3D              DB      '<='
02CE 85EB              DB      XP14 SHR 8 + 128,XP14 AND 255
02D0 3C                DB      '<'
02D1 85F9              DB      XP16 SHR 8 + 128,XP16 AND 255
02D3 85FF              DB      XP17 SHR 8 + 128,XP17 AND 255
                ;*
02D5 21F801     DIRECT LXI     H,TAB1-1        ;*** DIRECT ***
                ;*
02D8 =          EXEC   EQU     $       ;*** EXEC ***
02D8 EF         EX0    RST     5       ;IGNORE LEADING BLANKS
02D9 D5                PUSH    D       ;SAVE POINTER
02DA 1A         EX1    LDAX    D       ;IFF FOUND '.' IN STRING
02DB 13                INX     D       ;BEFORE ANY MISMATCH
02DC FE2E              CPI     56Q     ;WE DECLARE A MATCH
02DE CAF702            JZ      EX3
02E1 23                INX     H       ;HL->TABLE
02E2 BE                CMP     M       ;IFF MATCH, TEST NEXT
02E3 CADA02            JZ      EX1
02E6 3E7F              MVI     A,177Q  ;ELSE, SEE IFF BIT 7
02E8 1B                DCX     D       ;OF TABLEIS SET, WHICH
02E9 BE                CMP     M       ;IS THE JUMP ADDR. (HI)
02EA DAFE02            JC      EX5     ;C:YES, MATCHED
02ED 23         EX2    INX     H       ;NC:NO, FIND JUMP ADDR.
02EE BE                CMP     M
02EF D2ED02            JNC     EX2
02F2 23                INX     H       ;BUMP TO NEXT TAB. ITEM
02F3 D1                POP     D       ;RESTORE STRING POINTER
02F4 C3D802            JMP     EX0     ;TEST AGAINST NEXT ITEM
02F7 3E7F       EX3    MVI     A,177Q  ;PARTIAL MATCH, FIND
02F9 23         EX4    INX     H       ;JUMP ADDR., WHICH IS
02FA BE                CMP     M       ;FLAGGED BY BIT 7
02FB D2F902            JNC     EX4
02FE 7E         EX5    MOV     A,M     ;LOAD HL WITH THE JUMP
02FF 23                INX     H       ;ADDRESS FROM THE TABLE
0300 6E                MOV     L,M
0301 E67F              ANI     177Q    ;MASK OFF BIT 7
0303 67                MOV     H,A
0304 F1                POP     PSW     ;CLEAN UP THE GABAGE
0305 E9                PCHL            ;AND WE GO DO IT
                ;*
                ;*************************************************************
                ;*
                ;* WHAT FOLLOWS IS THE CODE TO EXECUTE DIRECT AND STATEMENT
                ;* COMMANDS. CONTROL IS TRANSFERED TO THESE POINTS VIA THE
                ;* COMMAND TABLE LOOKUP CODE OF 'DIRECT' AND 'EXEC' IN LAST
                ;* SECTION. AFTER THE COMMAND IS EXECUTED, CONTROL IS
                ;* TANSFERED TO OTHER SECTIONS AS FOLLOWS:
                ;*
                ;* FOR 'LIST', 'NEW', AND 'STOP': GO BACK TO 'RSTART'
                ;* FOR 'RUN': GO EXECUTE THE FIRST STORED LINE IFF ANY; ELSE
                ;* GO BACK TO 'RSTART'.
                ;* FOR 'GOTO' AND 'GOSUB': GO EXECUTE THE TARGET LINE.
                ;* FOR 'RETURN' AND 'NEXT': GO BACK TO SAVED RETURN LINE.
                ;* FOR ALL OTHERS: IFF 'CURRNT' -> 0, GO TO 'RSTART', ELSE
                ;* GO EXECUTE NEXT COMMAND. (THIS IS DONE IN 'FINISH'.)
                ;*
                ;*************************************************************
                ;*
                ;* *** NEW *** STOP *** RUN (& FRIENDS) *** & GOTO ***
                ;*
```

```
                ;* 'NEW(CR)' SETS 'TXTUNF' TO POINT TO 'TXTBGN'
                ;*
                ;* 'STOP(CR)' GOES BACK TO 'RSTART'
                ;*
                ;* 'RUN(CR)' FINDS THE FIRST STORED LINE, STORE ITS ADDRESS (IN
                ;* 'CURRNT'), AND START EXECUTE IT. NOTE THAT ONLY THOSE
                ;* COMMANDS IN TAB2 ARE LEGAL FOR STORED PROGRAM.
                ;*
                ;* THERE ARE 3 MORE ENTRIES IN 'RUN':
                ;* 'RUNNXL' FINDS NEXT LINE, STORES ITS ADDR. AND EXECUTES IT.
                ;* 'RUNTSL' STORES THE ADDRESS OF THIS LINE AND EXECUTES IT.
                ;* 'RUNSML' CONTINUES THE EXECUTION ON SAME LINE.;*
                ;* 'GOTO EXPR(CR)' EVALUATES THE EXPRESSION, FIND THE TARGET
                ;* LINE, AND JUMP TO 'RUNTSL' TO DO IT.
                ;* 'DLOAD' LOADS A NAMED PROGRAM FROM DISK.
                ;* 'DSAVE' SAVES A NAMED PROGRAM ON DISK.
                ;* 'FCBSET' SETS UP THE FILE CONTROL BLOCK FOR SUBSEQUENT DISK I/O
                ;*
0306 CDDC07     NEW     CALL    ENDCHK  ;*** NEW(CR) ***
0309 21D709             LXI     H,TXTBGN
030C 22D509             SHLD    TXTUNF
                ;*
030F CDDC07     STOP    CALL    ENDCHK  ;*** STOP(CR) ***
0312 C38101             JMP     RSTART
                ;*
0315 CDDC07     RUN     CALL    ENDCHK  ;*** RUN(CR) ***
0318 11D709             LXI     D,TXTBGN ;FIRST SAVED LINE
                ;*
031B 210000     RUNNXL  LXI     H,0     ;*** RUNNXL ***
031E CD5F08             CALL    FNDLNP  ;FIND WHATEVER LINE #
0321 DA8101             JC      RSTART  ;C:PASSED TXTUNF, QUIT
                ;*
0324 EB         RUNTSL  XCHG            ;*** RUNTSL ***
0325 22C109             SHLD    CURRNT  ;SET 'CURRNT'->LINE #
0328 EB                 XCHG
0329 13                 INX     D       ;BUMP PASS LINE #
032A 13                 INX     D
                ;*
032B CD8509     RUNSML  CALL    CHKIO   ;*** RUNSML ***
032E 211902             LXI     H,TAB2-1 ;FIND COMMAND IN TAB2
0331 C3D802             JMP     EXEC    ;AND EXECUTE IT
                ;*
0334 DF         GOTO    RST     3       ;*** GOTO EXPR ***
0335 D5                 PUSH    D       ;SAVE FOR ERROR ROUTINE
0336 CDDC07             CALL    ENDCHK  ;MUST FIND A 0DH
0339 CD5708             CALL    FNDLN   ;FIND THE TARGET LINE
033C C26701             JNZ     AHOW    ;NO SUCH LINE #
033F F1                 POP     PSW     ;CLEAR THE "PUSH  DE"
0340 C32403             JMP     RUNTSL  ;GO DO IT
0005 =          CPM     EQU     5       ;DISK PARAMETERS
005C =          FCB     EQU     5CH
001A =          SETDMA  EQU     26
000F =          OPEN    EQU     15
0014 =          READD   EQU     20
0015 =          WRITED  EQU     21
0010 =          CLOSE   EQU     16
0016 =          MAKE    EQU     22
0013 =          DELETE  EQU     19
                ;*
0343 EF         DLOAD   RST     5       ;IGNORE BLANKS
0344 E5                 PUSH    H       ;SAVE H
0345 CDE503             CALL    FCBSET  ;SET UP FILE CONTROL BLOCK
0348 D5                 PUSH    D       ;SAVE THE REST
0349 C5                 PUSH    B
034A 115C00             LXI     D,FCB   ;GET FCB ADDRESS
034D 0E0F               MVI     C,OPEN  ;PREPARE TO OPEN FILE
034F CD0500             CALL    CPM     ;OPEN IT
0352 FEFF               CPI     0FFH    ;IS IT THERE?
0354 CA6601             JZ      QHOW    ;NO, SEND ERROR
0357 AF                 XRA     A       ;CLEAR A
0358 327C00             STA     FCB+32  ;START AT RECORD 0
035B 11D509             LXI     D,TXTUNF ;GET BEGINNING
035E D5         LOAD    PUSH    D       ;SAVE DMA ADDRESS
035F 0E1A               MVI     C,SETDMA        ;
0361 CD0500             CALL    CPM     ;SET DMA ADDRESS
0364 0E14               MVI     C,READD ;
0366 115C00             LXI     D,FCB
0369 CD0500             CALL    CPM     ;READ SECTOR
036C FE01               CPI     1       ;DONE?
036E DA8103             JC      RDMORE  ;NO, READ MORE
0371 C26601             JNZ     QHOW    ;BAD READ
```

```
0374 0E10            MVI     C,CLOSE
0376 115C00          LXI     D,FCB
0379 CD0500          CALL    CPM       ;CLOSE FILE
037C D1              POP     D         ;THROW AWAY DMA ADD.
037D C1              POP     B         ;GET OLD REGISTERS BACK
037E D1              POP     D
037F E1              POP     H
0380 F7              RST     6         ;FINISH
0381 D1      RDMORE  POP     D         ;GET DMA ADDRESS
0382 218000          LXI     H,80H     ;GET 128
0385 19              DAD     D         ;ADD 128 TO DMA ADD.
0386 EB              XCHG              ;PUT IT BACK IN D
0387 C35E03          JMP     LOAD      ;AND READ SOME MORE
             ;*
038A EF      DSAVE   RST     5         ;IGNORE BLANKS
038B E5              PUSH    H         ;SAVE H
038C CDE503          CALL    FCBSET    ;SETUP FCB
038F D5              PUSH    D
0390 C5              PUSH    B         ;SAVE OTHERS
0391 115C00          LXI     D,FCB
0394 0E13            MVI     C,DELETE
0396 CD0500          CALL    CPM       ;ERASE FILE IF IT EXISTS
0399 115C00          LXI     D,FCB
039C 0E16            MVI     C,MAKE
039E CD0500          CALL    CPM       ;MAKE A NEW ONE
03A1 FEFF            CPI     0FFH      ;IS THERE SPACE?
03A3 CA6601          JZ      QHOW      ;NO, ERROR
03A6 AF              XRA     A         ;CLEAR A
03A7 327C00          STA     FCB+32    ;START AT RECORD 0
03AA 11D509          LXI     D,TXTUNF  ;GET BEGINNING
03AD D5      SAVE    PUSH    D         ;SAVE DMA ADDRESS
03AE 0E1A            MVI     C,SETDMA          ;
03B0 CD0500          CALL    CPM       ;SET DMA ADDRESS
03B3 0E15            MVI     C,WRITED
03B5 115C00          LXI     D,FCB
03B8 CD0500          CALL    CPM       ;WRITE SECTOR
03BB B7              ORA     A         ;SET FLAGS
03BC C26601          JNZ     QHOW      ;IF NOT ZERO, ERROR
03BF D1              POP     D         ;GET DMA ADD. BACK
03C0 3AD609          LDA     TXTUNF+1  ;AND MSB OF LAST ADD.
03C3 BA              CMP     D         ;IS D SMALLER?
03C4 DAD903          JC      SAVDON    ;YES, DONE
03C7 C2D103          JNZ     WRITMOR   ;DONT TEST E IF NOT EQUAL
03CA 3AD509          LDA     TXTUNF    ;IS E SMALLER?
03CD BB              CMP     E
03CE DAD903          JC      SAVDON    ;YES, DONE
03D1 218000  WRITMOR LXI     H,80H
03D4 19              DAD     D         ;ADD 128 TO DMA ADD.
03D5 EB              XCHG              ;GET IT BACK IN D
03D6 C3AD03          JMP     SAVE      ;WRITE SOME MORE
03D9 0E10    SAVDON  MVI     C,CLOSE
03DB 115C00          LXI     D,FCB
03DE CD0500          CALL    CPM       ;CLOSE FILE
03E1 C1              POP     B         ;GET REGISTERS BACK
03E2 D1              POP     D
03E3 E1              POP     H
03E4 F7              RST     6         ;FINISH
             ;*
03E5 215C00  FCBSET  LXI     H,FCB     ;GET FILE CONTROL BLOCK ADDRESS
03E8 3600            MVI     M,0       ;CLEAR ENTRY TYPE
03EA 23      FNCLR   INX     H         ;NEXT LOCATION
03EB 3620            MVI     M,' '     ;CLEAR TO SPACE
03ED 3E64            MVI     A,FCB+8 AND 255
03EF BD              CMP     L         ;DONE?
03F0 C2EA03          JNZ     FNCLR     ;NO, DO IT AGAIN
03F3 23              INX     H         ;NEXT
03F4 3654            MVI     M,'T'     ;SET FILE TYPE TO 'TBI'
03F6 23              INX     H
03F7 3642            MVI     M,'B'
03F9 23              INX     H
03FA 3649            MVI     M,'I'
03FC 23      EXRC    INX     H         ;CLEAR REST OF FCB
03FD 3600            MVI     M,0
03FF 3E6B            MVI     A,FCB+15 AND 255
0401 BD              CMP     L         ;DONE?
0402 C2FC03          JNZ     EXRC      ;NO, CONTINUE
0405 215D00          LXI     H,FCB+1   ;GET FILENAME START
0408 1A      FN      LDAX    D         ;GET CHARACTER
0409 FE0D            CPI     0DH       ;IS IT A 'CR'
040B C8              RZ                ;YES, DONE
040C FE21            CPI     '!'       ;LEGAL CHARACTER?
040E DAE007          JC      QWHAT     ;NO, SEND ERROR
```

```
0411 FE5B             CPI     'Z'       ;AGAIN
0413 D2E007           JNC     QWHAT     ;DITTO
0416 77               MOV     M,A       ;SAVE IT IN FCB
0417 23               INX     H         ;NEXT
0418 13               INX     D
0419 3E65             MVI     A,FCB+9 AND 255
041B BD               CMP     L         ;LAST?
041C C20804           JNZ     FN        ;NO, CONTINUE
041F C9               RET               ;TRUNCATE AT 8 CHARACTERS
                ;*
                ;*************************************************************
                ;*
                ;* *** LIST *** & PRINT ***
                ;*
                ;* LIST HAS TWO FORMS:
                ;* 'LIST(CR)' LISTS ALL SAVED LINES
                ;* 'LIST #(CR)' START LIST AT THIS LINE #
                ;* YOU CAN STOP THE LISTING BY CONTROL C KEY
                ;*
                ;* PRINT COMMAND IS 'PRINT ....;' OR 'PRINT ....(CR)'
                ;* WHERE '....' IS A LIST OF EXPRESIONS, FORMATS, BACK-
                ;* ARROWS, AND STRINGS. THESE ITEMS ARE SEPERATED BY COMMAS.
                ;*
                ;* A FORMAT IS A POUND SIGN FOLLOWED BY A NUMBER. IT CONTROLSS
                ;* THE NUMBER OF SPACES THE VALUE OF A EXPRESION IS GOING TO
                ;* BE PRINTED.  IT STAYS EFFECTIVE FOR THE REST OF THE PRINT
                ;* COMMAND UNLESS CHANGED BY ANOTHER FORMAT. IFF NO FORMAT IS
                ;* SPECIFIED, 6 POSITIONS WILL BE USED.
                ;*
                ;* A STRING IS QUOTED IN A PAIR OF SINGLE QUOTES OR A PAIR OF
                ;* DOUBLE QUOTES.
                ;*
                ;* A BACK-ARROW MEANS GENERATE A (CR) WITHOUT (LF)
                ;*
                ;* A (CRLF) IS GENERATED AFTER THE ENTIRE LIST HAS BEEN
                ;* PRINTED OR IFF THE LIST IS A NULL LIST. HOWEVER IFF THE LIST
                ;* ENDED WITH A COMMA, NO (CRL) IS GENERATED.
                ;*
0420 CD3E01     LIST    CALL    TSTNUM    ;TEST IFF THERE IS A #
0423 CDDC07             CALL    ENDCHK    ;IFF NO # WE GET A 0
0426 CD5708             CALL    FNDLN     ;FIND THIS OR NEXT LINE
0429 DA8101     LS1     JC      RSTART    ;C:PASSED TXTUNF
042C CDED08             CALL    PRTLN     ;PRINT THE LINE
042F CD8509             CALL    CHKIO     ;STOP IFF HIT CONTROL-C
0432 CD5F08             CALL    FNDLNP    ;FIND NEXT LINE
0435 C32904             JMP     LS1       ;AND LOOP BACK
                ;*
0438 0E06       PRINT   MVI     C,6       ;C = # OF SPACES
043A CF                 RST     1         ;IFF NULL LIST & ";"
043B 3B                 DB      73Q
043C 06                 DB      6Q
043D CD0E00             CALL    CRLF      ;GIVE CR-LF AND
0440 C32B03             JMP     RUNSML    ;CONTINUE SAME LINE
0443 CF         PR2     RST     1         ;IFF NULL LIST (CR)
0444 0D                 DB      0DH
0445 06                 DB      6Q
0446 CD0E00             CALL    CRLF      ;ALSO GIVE CR-LF AND
0449 C31B03             JMP     RUNNXL    ;GO TO NEXT LINE
044C CF         PR0     RST     1         ;ELSE IS IT FORMAT?
044D 23                 DB      '#'
044E 05                 DB      5Q
044F DF                 RST     3         ;YES, EVALUATE EXPR.
0450 4D                 MOV     C,L       ;AND SAVE IT IN C
0451 C35A04             JMP     PR3       ;LOOK FOR MORE TO PRINT
0454 CD8B08     PR1     CALL    QTSTG     ;OR IS IT A STRING?
0457 C36704             JMP     PR8       ;IFF NOT, MUST BE EXPR.
045A CF         PR3     RST     1         ;IFF ",", GO FIND NEXT
045B 2C                 DB      ','
045C 06                 DB      6Q
045D CDCD07             CALL    FIN       ;IN THE LIST.
0460 C34C04             JMP     PR0       ;LIST CONTINUES
0463 CD0E00     PR6     CALL    CRLF      ;LIST ENDS
0466 F7                 RST     6
0467 DF         PR8     RST     3         ;EVALUATE THE EXPR
0468 C5                 PUSH    B
0469 CD8108             CALL    PRTNUM    ;PRINT THE VALUE
046C C1                 POP     B
046D C35A04             JMP     PR3       ;MORE TO PRINT?
                ;*
                ;*************************************************************
                ;*
                ;* *** GOSUB *** & RETURN ***
```

```
                        ;*
                        ;* 'GOSUB EXPR;' OR 'GOSUB EXPR (CR)' IS LIKE THE 'GOTO'
                        ;* COMMAND, EXCEPT THAT THE CURRENT TEXT POINTER, STACK POINTER
                        ;* ETC. ARE SAVE SO THAT EXECUTION CAN BE CONTINUED AFTER THE
                        ;* SUBROUTINE 'RETURN'. IN ORDER THAT 'GOSUB' CAN BE NESTED
                        ;* (AND EVEN RECURSIVE), THE SAVE AREA MUST BE STACKED.
                        ;* THE STACK POINTER IS SAVED IN 'STKGOS'. THE OLD 'STKGOS' IS
                        ;* SAVED IN THE STACK.  IFF WE ARE IN THE MAIN ROUTINE, 'STKGOS
                        ;* IS ZERO (THIS WAS DONE BY THE "MAIN" SECTION OF THE CODE),
                        ;* BUT WE STILL SAVE IT AS A FLAG FOR NO FURTHER RETURNS.
                        ;*
                        ;* 'RETURN(CR)' UNDOS EVERYHING THAT 'GOSUB' DID, AND THUS
                        ;* RETURN THE EXCUTION TO THE COMMAND AFTER THE MOST RECENT
                        ;* 'GOSUB'. IFF 'STKGOS' IS ZERO, IT INDICATES THAT WE
                        ;* NEVER HAD A 'GOSUB' AND IS THUS AN ERROR.
                        ;*
0470 CD3409     GOSUB   CALL    PUSHA   ;SAVE THE CURRENT "FOR"
0473 DF                 RST     3       ;PARAMETERS
0474 D5                 PUSH    D       ;AND TEXT POINTER
0475 CD5708             CALL    FNDLN   ;FIND THE TARGET LINE
0478 C26701             JNZ     AHOW    ;NOT THERE. SAY "HOW?"
047B 2AC109             LHLD    CURRNT  ;FOUND IT, SAVE OLD
047E E5                 PUSH    H       ;'CURRNT' OLD 'STKGOS'
047F 2AC309             LHLD    STKGOS
0482 E5                 PUSH    H
0483 210000             LXI     H,0     ;AND LOAD NEW ONES
0486 22C909             SHLD    LOPVAR
0489 39                 DAD     SP
048A 22C309             SHLD    STKGOS
048D C32403             JMP     RUNTSL  ;THEN RUN THAT LINE
0490 CDDC07     RETURN  CALL    ENDCHK  ;THERE MUST BE A ODH
0493 2AC309             LHLD    STKGOS  ;OLD STACK POINTER
0496 7C                 MOV     A,H     ;0 MEANS NOT EXIST
0497 B5                 ORA     L
0498 CAE007             JZ      QWHAT   ;SO, WE SAY: "WHAT?"
049B F9                 SPHL            ;ELSE, RESTORE IT
049C E1                 POP     H
049D 22C309             SHLD    STKGOS  ;AND THE OLD 'STKGOS'
04A0 E1                 POP     H
04A1 22C109             SHLD    CURRNT  ;AND THE OLD 'CURRNT'
04A4 D1                 POP     D       ;OLD TEXT POINTER
04A5 CD1809             CALL    POPA    ;OLD "FOR" PARAMETERS
04A8 F7                 RST     6       ;AND WE ARE BACK HOME
                        ;*
                        ;*****************************************************************
                        ;*
                        ;* *** FOR *** & NEXT ***
                        ;*
                        ;* 'FOR' HAS TWO FORMS:
                        ;* 'FOR VAR=EXP1 TO EXP2 STEP EXP1' AND 'FOR VAR=EXP1 TO EXP2'
                        ;* THE SECOND FORM MEANS THE SAME THING AS THE FIRST FORM WITH
                        ;* EXP1=1. (I.E., WITH A STEP OF +1.)
                        ;* TBI WILL FIND THE VARIABLE VAR. AND SET ITS VALUE TO THE
                        ;* CURRENT VALUE OF EXP1. IT ALSO EVALUATES EXPR2 AND EXP1
                        ;* AND SAVE ALL THESE TOGETHER WITH THE TEXT POINTERR ETC. IN
                        ;* THE 'FOR' SAVE AREA, WHICH CONSISTS OF 'LOPVAR', 'LOPINC',
                        ;* 'LOPLMT', 'LOPLN', AND 'LOPPT'. IFF THERE IS ALREADY SOME-
                        ;* THING IN THE SAVE AREA (THIS IS INDICATED BY A NON-ZERO
                        ;* 'LOPVAR'), THEN THE OLD SAVE AREA IS SAVED IN THE STACK
                        ;* BEFORE THE NEW ONE OVERWRITES IT.
                        ;* TBI WILL THEN DIG IN THE STACK AND FIND OUT IFF THIS SAME
                        ;* VARIABLE WAS USED IN ANOTHER CURRENTLY ACTIVE 'FOR' LOOP.
                        ;* IFF THAT IS THE CASE THEN THE OLD 'FOR' LOOP IS DEACTIVATED.
                        ;* (PURGED FROM THE STACK..)
                        ;*
                        ;* 'NEXT VAR' SERVES AS THE LOGICAL (NOT NECESSARILLY PHYSICAL)
                        ;* END OF THE 'FOR' LOOP. THE CONTROL VARIABLE VAR. IS CHECKED
                        ;* WITH THE 'LOPVAR'.   IFF THEY ARE NOT THE SAME, TBI DIGS IN
                        ;* THE STACK TO FIND THE RIGHTT ONE AND PURGES ALL THOSE THAT
                        ;* DID NOT MATCH. EITHER WAY, TBI THEN ADDS THE 'STEP' TO
                        ;* THAT VARIABLE AND CHECK THE RESULT WITH THE LIMIT. IFF IT
                        ;* IS WITHIN THE LIMIT, CONTROL LOOPS BACK TO THE COMMAND
                        ;* FOLLOWING THE 'FOR'. IFF OUTSIDE THE LIMIT, THE SAVE AREA
                        ;* IS PURGED AND EXECUTION CONTINUES.
                        ;*
04A9 CD3409     FOR     CALL    PUSHA   ;SAVE THE OLD SAVE AREA
04AC CDBA07             CALL    SETVAL  ;SET THE CONTROL VAR.
04AF 2B                 DCX     H       ;HL IS ITS ADDRESS
04B0 22C909             SHLD    LOPVAR  ;SAVE THAT
04B3 21B002             LXI     H,TAB5-1 ;USE 'EXEC' TO LOOK
04B6 C3B802             JMP     EXEC    ;FOR THE WORD 'TO'
04B9 DF         FR1     RST     3       ;EVALUATE THE LIMIT
```

```
04BA 22CD09            SHLD    LOPLMT  ;SAVE THAT
04BD 21B602            LXI     H,TAB6-1 ;USE 'EXEC' TO LOOK
04C0 C3D802            JMP     EXEC    ;FOR THE WORD 'STEP'
04C3 DF        FR2     RST     3       ;FOUND IT, GET STEP
04C4 C3CA04            JMP     FR4
04C7 210100    FR3     LXI     H,1Q    ;NOT FOUND, SET TO 1
04CA 22CB09    FR4     SHLD    LOPINC  ;SAVE THAT TOO
04CD 2AC109    FR5     LHLD    CURRNT  ;SAVE CURRENT LINE #
04D0 22CF09            SHLD    LOPLN
04D3 EB                XCHG            ;AND TEXT POINTER
04D4 22D109            SHLD    LOPPT
04D7 010A00            LXI     B,12Q   ;DIG INTO STACK TO
04DA 2AC909            LHLD    LOPVAR  ;FIND 'LOPVAR'
04DD EB                XCHG
04DE 60                MOV     H,B
04DF 68                MOV     L,B     ;HL=0 NOW
04E0 39                DAD     SP      ;HERE IS THE STACK
04E1 3E                DB      76Q
04E2 09        FR7     DAD     B       ;EACH LEVEL IS 10 DEEP
04E3 7E                MOV     A,M     ;GET THAT OLD 'LOPVAR'
04E4 23                INX     H
04E5 B6                ORA     M
04E6 CA0305            JZ      FR8     ;0 SAYS NO MORE IN IT
04E9 7E                MOV     A,M
04EA 2B                DCX     H
04EB BA                CMP     D       ;SAME AS THIS ONE?
04EC C2E204            JNZ     FR7
04EF 7E                MOV     A,M     ;THE OTHER HALF?
04F0 BB                CMP     E
04F1 C2E204            JNZ     FR7
04F4 EB                XCHG            ;YES, FOUND ONE
04F5 210000            LXI     H,0
04F8 39                DAD     SP      ;TRY TO MOVE SP
04F9 44                MOV     B,H
04FA 4D                MOV     C,L
04FB 210A00            LXI     H,12Q
04FE 19                DAD     D
04FF CD0909            CALL    MVDOWN  ;AND PURGE 10 WORDS
0502 F9                SPHL            ;IN THE STACK
0503 2AD109    FR8     LHLD    LOPPT   ;JOB DONE, RESTORE DE
0506 EB                XCHG
0507 F7                RST     6       ;AND CONTINUE
                ;*
0508 FF        NEXT    RST     7       ;GET ADDRESS OF VAR.
0509 DAE007            JC      QWHAT   ;NO VARIABLE, "WHAT?"
050C 22C509            SHLD    VARNXT  ;YES, SAVE IT
050F D5        NX0     PUSH    D       ;SAVE TEXT POINTER
0510 EB                XCHG
0511 2AC909            LHLD    LOPVAR  ;GET VAR. IN 'FOR'
0514 7C                MOV     A,H
0515 B5                ORA     L       ;0 SAYS NEVER HAD ONE
0516 CAE107            JZ      AWHAT   ;SO WE ASK: "WHAT?"
0519 E7                RST     4       ;ELSE WE CHECK THEM
051A CA2705            JZ      NX3     ;OK, THEY AGREE
051D D1                POP     D       ;NO, LET'S SEE
051E CD1809            CALL    POPA    ;PURGE CURRENT LOOP
0521 2AC509            LHLD    VARNXT  ;AND POP ONE LEVEL
0524 C30F05            JMP     NX0     ;GO CHECK AGAIN
0527 5E        NX3     MOV     E,M     ;COME HERE WHEN AGREED
0528 23                INX     H
0529 56                MOV     D,M     ;DE=VALUE OF VAR.
052A 2ACB09            LHLD    LOPINC
052D E5                PUSH    H
052E 19                DAD     D       ;ADD ONE STEP
052F EB                XCHG
0530 2AC909            LHLD    LOPVAR  ;PUT IT BACK
0533 73                MOV     M,E
0534 23                INX     H
0535 72                MOV     M,D
0536 2ACD09            LHLD    LOPLMT  ;HL->LIMIT
0539 F1                POP     PSW     ;OLD HL
053A B7                ORA     A
053B F23F05            JP      NX1     ;STEP > 0
053E EB                XCHG
053F CDB207    NX1     CALL    CKHLDE  ;COMPARE WITH LIMIT
0542 D1                POP     D       ;RESTORE TEXT POINTER
0543 DA5105            JC      NX2     ;OUTSIDE LIMIT
0546 2ACF09            LHLD    LOPLN   ;WITHIN LIMIT, GO
0549 22C109            SHLD    CURRNT  ;BACK TO THE SAVED
054C 2AD109            LHLD    LOPPT   ;'CURRNT' AND TEXT
054F EB                XCHG            ;POINTER
0550 F7                RST     6
```

```
0551 CD1809     NX2     CALL    POPA    ;PURGE THIS LOOP
0554 F7                 RST     6
                ;*
                ;*****************************************************************
                ;*
                ;* *** REM *** IFF *** INPUT *** & LET (& DEFLT) ***
                ;*
                ;* 'REM' CAN BE FOLLOWED BY ANYTHING AND IS IGNORED BY TBI.
                ;* TBI TREATS IT LIKE AN 'IF' WITH A FALSE CONDITION.
                ;*
                ;* 'IF' IS FOLLOWED BY AN EXPR. AS A CONDITION AND ONE OR MORE
                ;* COMMANDS (INCLUDING OUTHER 'IF'S) SEPERATED BY SEMI-COLONS.
                ;* NOTE THAT THE WORD 'THEN' IS NOT USED. TBI EVALUATES THE
                ;* EXPR. IFF IT IS NON-ZERO, EXECUTION CONTINUES. IFF THE
                ;* EXPR. IS ZERO, THE COMMANDS THAT FOLLOWS ARE IGNORED AND
                ;* EXECUTION CONTINUES AT THE NEXT LINE.
                ;*
                ;* 'IPUT' COMMAND IS LIKE THE 'PRINT' COMMAND, AND IS FOLLOWED
                ;* BY A LIST OF ITEMS.  IFF THE ITEM IS A STRING IN SINGLE OR
                ;* DOUBLE QUOTES, OR IS A BACK-ARROW, IT HAS THE SAME EFFECT AS
                ;* IN 'PRINT'.  IFF AN ITEM IS A VARIABLE, THIS VARIABLE NAME IS
                ;* PRINTED OUT FOLLOWED BY A COLON. THEN TBI WAITS FOR AN
                ;* EXPR. TO BE TYPED IN. THE VARIABLE ISS THEN SET TO THE
                ;* VALUE OF THIS EXPR.  IFF THE VARIABLE IS PROCEDED BY A STRING
                ;* (AGAIN IN SINGLE OR DOUBLE QUOTES), THE STRING WILL BE
                ;* PRINTED FOLLOWED BY A COLON. TBI THEN WAITS FOR INPUT EXPR.
                ;* AND SET THE VARIABLE TO THE VALUE OF THE EXPR.
                ;*
                ;* IFF THE INPUT EXPR. IS INVALID, TBI WILL PRINT "WHAT?",
                ;* "HOW?" OR "SORRY" AND REPRINT THE PROMPT AND REDO THE INPUT.
                ;* THE EXECUTION WILL NOT TERMINATE UNLESS YOU TYPE CONTROL-C.
                ;* THIS IS HANDLED IN 'INPERR'.
                ;*
                ;* 'LET' IS FOLLOWED BY A LIST OF ITEMS SEPERATED BY COMMAS.
                ;* EACH ITEM CONSISTS OF A VARIABLE, AN EQUAL SIGN, AND AN EXPR.
                ;* TBI EVALUATES THE EXPR. AND SET THE VARIBLE TO THAT VALUE.
                ;* TB WILL ALSO HANDLE 'LET' COMMAND WITHOUT THE WORD 'LET'.
                ;* THIS IS DONE BY 'DEFLT'.
                ;*
0555 210000     REM     LXI     H,0Q    ;*** REM ***
0558 3E                 DB      76Q
                ;*
0559 DF         IFF     RST     3       ;*** IFF ***
055A 7C                 MOV     A,H     ;IS THE EXPR.=0?
055B B5                 ORA     L
055C C22B03             JNZ     RUNSML  ;NO, CONTINUE
055F CD7508             CALL    FNDSKP  ;YES, SKIP REST OF LINE
0562 D22403             JNC     RUNTSL
0565 C38101             JMP     RSTART
                ;*
0568 2AC709     INPERR  LHLD    STKINP  ;*** INPERR ***
056B F9                 SPHL            ;RESTORE OLD SP
056C E1                 POP     H       ;AND OLD 'CURRNT'
056D 22C109             SHLD    CURRNT
0570 D1                 POP     D       ;AND OLD TEXT POINTER
0571 D1                 POP     D       ;REDO INPUT
                ;*
0572 =          INPUT   EQU     $       ;*** INPUT ***
0572 D5         IP1     PUSH    D       ;SAVE IN CASE OF ERROR
0573 CD8B08             CALL    QTSTG   ;IS NEXT ITEM A STRING?
0576 C38005             JMP     IP2     ;NO
0579 FF                 RST     7       ;YES. BUT FOLLOWED BY A
057A DABA05             JC      IP4     ;VARIABLE?      NO.
057D C39005             JMP     IP3     ;YES.   INPUT VARIABLE
0580 D5         IP2     PUSH    D       ;SAVE FOR 'PRTSTG'
0581 FF                 RST     7       ;MUST BE VARIABLE NOW
0582 DAE007             JC      QWHAT   ;"WHAT?" IT IS NOT?
0585 1A                 LDAX    D       ;GET READY FOR 'RTSTG'
0586 4F                 MOV     C,A
0587 97                 SUB     A
0588 12                 STAX    D
0589 D1                 POP     D
058A CD7F08             CALL    PRTSTG  ;PRINT STRING AS PROMPT
058D 79                 MOV     A,C     ;RESTORE TEXT
058E 1B                 DCX     D
058F 12                 STAX    D
0590 D5         IP3     PUSH    D       ;SAVE IN CASE OF ERROR
0591 EB                 XCHG
0592 2AC109             LHLD    CURRNT  ;ALSO SAVE 'CURRNT'
0595 E5                 PUSH    H
0596 217205             LXI     H,IP1   ;A NEGATIVE NUMBER
0599 22C109             SHLD    CURRNT  ;AS A FLAG
```

```
059C 210000             LXI     H,0Q    ;SAVE SP TOO
059F 39                 DAD     SP
05A0 22C709             SHLD    STKINP
05A3 D5                 PUSH    D       ;OLD HL
05A4 3E3A               MVI     A,72Q   ;PRINT THIS TOO
05A6 CD1408             CALL    GETLN   ;AND GET A LINE
05A9 11370F     IP3A    LXI     D,BUFFER ;POINTS TO BUFFER
05AC DF                 RST     3       ;EVALUATE INPUT
05AD 00                 NOP             ;CAN BE 'CALL   ENDCHK'
05AE 00                 NOP
05AF 00                 NOP
05B0 D1                 POP     D       ;OK, GET OLD HL
05B1 EB                 XCHG
05B2 73                 MOV     M,E     ;SAVE VALUE IN VAR.
05B3 23                 INX     H
05B4 72                 MOV     M,D
05B5 E1                 POP     H       ;GET OLD 'CURRNT'
05B6 22C109             SHLD    CURRNT
05B9 D1                 POP     D       ;AND OLD TEXT POINTER
05BA F1         IP4     POP     PSW     ;PURGE JUNK IN STACK
05BB CF                 RST     1       ;IS NEXT CH. ','?
05BC 2C                 DB      ','
05BD 03                 DB      3Q
05BE C37205             JMP     IP1     ;YES, MORE ITEMS.
05C1 F7         IP5     RST     6
                ;*
05C2 1A         DEFLT   LDAX    D       ;*** DEFLT ***
05C3 FE0D               CPI     0DH     ;EMPTY LINE IS OK
05C5 CAD105             JZ      LT1     ;ELSE IT IS 'LET'
                ;*
05C8 CDBA07     LET     CALL    SETVAL  ;*** LET ***
05CB CF                 RST     1       ;SET VALUE TO VAR.
05CC 2C                 DB      ','
05CD 03                 DB      3Q
05CE C3C805             JMP     LET     ;ITEM BY ITEM
05D1 F7         LT1     RST     6       ;UNTIL FINISH
                ;*
                ;*************************************************************
                ;*
                ;* *** EXPR ***
                ;*
                ;* 'EXPR' EVALUATES ARITHMETICAL OR LOGICAL EXPRESSIONS.
                ;* <EXPR>::=<EXPR2>
                ;*                         <EXPR2><REL.OP.><EXPR2>
                ;* WHERE <REL.OP.> IS ONE OF THE OPERATORSS IN TAB8 AND THE
                ;* RESULT OF THESE OPERATIONS IS 1 IFF TRUE AND 0 IFF FALSE.
                ;* <EXPR2>::=(+ OR -)<EXPR3>(+ OR -<EXPR3>)(....)
                ;* WHERE () ARE OPTIONAL AND (....) ARE OPTIONAL REPEATS.
                ;* <EXPR3>::=<EXPR4>(<* OR /><EXPR4>)(....)
                ;* <EXPR4>::=<VARIABLE>
                ;*                         <FUNCTION>
                ;*                         (<EXPR>)
                ;* <EXPR> IS RECURSIVE SO THAT VARIABLE '@' CAN HAVE AN <EXPR>
                ;* AS INDEX, FNCTIONS CAN HAVE AN <EXPR> AS ARGUMENTS, AND
                ;* <EXPR4> CAN BE AN <EXPR> IN PARANTHESE.
                ;*
                ;*      EXPR    CALL EXPR2      THIS IS AT LOC. 18
                ;*              PUSH    HL              SAVE <EXPR2> VALUE
05D2 21BE02     EXPR1   LXI     H,TAB8-1        ;LOOKUP REL.OP.
05D5 C3D802             JMP     EXEC    ;GO DO IT
05D8 CD0106     XP11    CALL    XP18    ;REL.OP.'>='
05DB D8                 RC              ;NO, RETURN HL=0
05DC 6F                 MOV     L,A     ;YES, RETURN HL=1
05DD C9                 RET
05DE CD0106     XP12    CALL    XP18    ;REL.OP.'#'
05E1 C8                 RZ              ;FALSE, RETURN HL=0
05E2 6F                 MOV     L,A     ;TRUE, RETURN HL=1
05E3 C9                 RET
05E4 CD0106     XP13    CALL    XP18    ;REL.OP.'>'
05E7 C8                 RZ              ;FALSE
05E8 D8                 RC              ;ALSO FALSE, HL=0
05E9 6F                 MOV     L,A     ;TRUE, HL=1
05EA C9                 RET
05EB CD0106     XP14    CALL    XP18    ;REL.OP.'<='
05EE 6F                 MOV     L,A     ;SET HL=1
05EF C8                 RZ              ;REL. TRUE, RETURN
05F0 D8                 RC
05F1 6C                 MOV     L,H     ;ELSE SET HL=0
05F2 C9                 RET
05F3 CD0106     XP15    CALL    XP18    ;REL.OP.'='
05F6 C0                 RNZ             ;FALSE, RETRUN HL=0
05F7 6F                 MOV     L,A     ;ELSE SET HL=1
```

```
05F8 C9                 RET
05F9 CD0106     XP16    CALL    XP18    ;REL.OP."<"
05FC D0                 RNC             ;FALSE, RETURN HL=0
05FD 6F                 MOV     L,A     ;ELSE SET HL=1
05FE C9                 RET
05FF E1         XP17    POP     H        ;NOT REL.OP.
0600 C9                 RET             ;RETURN HL=<EXPR2>
0601 79         XP18    MOV     A,C     ;SUBROUTINE FOR ALL
0602 E1                 POP     H       ;REL.OP.'S
0603 C1                 POP     B
0604 E5                 PUSH    H       ;REVERSE TOP OF STACK
0605 C5                 PUSH    B
0606 4F                 MOV     C,A
0607 CD1606             CALL    EXPR2   ;GET 2ND <EXPR2>
060A EB                 XCHG            ;VALUE IN DE NOW
060B E3                 XTHL            ;1ST <EXPR2> IN HL
060C CDB207             CALL    CKHLDE  ;COMPARE 1ST WITH 2ND
060F D1                 POP     D        ;RESTORE TEXT POINTER
0610 210000             LXI     H,0Q    ;SET HL=0, A=1
0613 3E01               MVI     A,1
0615 C9                 RET
                ;*
0616 CF         EXPR2   RST     1       ;NEGATIVE SIGN?
0617 2D                 DB      '-'
0618 06                 DB      6Q
0619 210000             LXI     H,0Q    ;YES, FAKE '0-'
061C C34006             JMP     XP26    ;TREAT LIKE SUBTRACT
061F CF         XP21    RST     1       ;POSITIVE SIGN? IGNORE
0620 2B                 DB      '+'
0621 00                 DB      0Q
0622 CD4A06     XP22    CALL    EXPR3   ;1ST <EXPR3>
0625 CF         XP23    RST     1       ;ADD?
0626 2B                 DB      '+'
0627 15                 DB      25Q
0628 E5                 PUSH    H       ;YES, SAVE VALUE
0629 CD4A06             CALL    EXPR3   ;GET 2ND<EXPR3>
062C EB         XP24    XCHG            ;2ND IN DE
062D E3                 XTHL            ;1ST IN HL
062E 7C                 MOV     A,H     ;COMPARE SIGN
062F AA                 XRA     D
0630 7A                 MOV     A,D
0631 19                 DAD     D
0632 D1                 POP     D       ;RESTORE TEXT POINTER
0633 FA2506             JM      XP23    ;1ST 2ND SIGN DIFFER
0636 AC                 XRA     H       ;1ST 2ND SIGN EQUAL
0637 F22506             JP      XP23    ;SO ISP RESULT
063A C36601             JMP     OHOW    ;ELSE WE HAVE OVERFLOW
063D CF         XP25    RST     1       ;SUBTRACT?
063E 2D                 DB      '-'
063F 83                 DB      203Q
0640 E5         XP26    PUSH    H       ;YES, SAVE 1ST <EXPR3>
0641 CD4A06             CALL    EXPR3   ;GET 2ND <EXPR3>
0644 CDA607             CALL    CHGSGN  ;NEGATE
0647 C32C06             JMP     XP24    ;AND ADD THEM
                ;*
064A CDA706     EXPR3   CALL    EXPR4   ;GET 1ST <EXPR4>
064D CF         XP31    RST     1       ;MULTIPLY?
064E 2A                 DB      '*'
064F 2C                 DB      54Q
0650 E5                 PUSH    H       ;YES, SAVE 1ST
0651 CDA706             CALL    EXPR4   ;AND GET 2ND <EXPR4>
0654 0600               MVI     B,0Q    ;CLEAR B FOR SIGN
0656 CDA307             CALL    CHKSGN  ;CHECK SIGN
0659 EB                 XCHG            ;2ND IN DE NOW
065A E3                 XTHL            ;1ST IN HL
065B CDA307             CALL    CHKSGN  ;CHECK SIGN OF 1ST
065E 7C                 MOV     A,H     ;IS HL > 255 ?
065F B7                 ORA     A
0660 CA6906             JZ      XP32    ;NO
0663 7A                 MOV     A,D     ;YES, HOW ABOUT DE
0664 B2                 ORA     D
0665 EB                 XCHG            ;PUT SMALLER IN HL
0666 C26701             JNZ     AHOW    ;ALSO >, WILL OVERFLOW
0669 7D         XP32    MOV     A,L     ;THIS IS DUMB
066A 210000             LXI     H,0Q    ;CLEAR RESULT
066D B7                 ORA     A       ;ADD AND COUNT
066E CA9906             JZ      XP35
0671 19         XP33    DAD     D
0672 DA6701             JC      AHOW    ;OVERFLOW
0675 3D                 DCR     A
0676 C27106             JNZ     XP33
0679 C39906             JMP     XP35    ;FINISHED
```

```
067C CF         XP34    RST     1       ;DIVIDE?
067D 2F                 DB      '/'
067E 44                 DB      104Q
067F E5                 PUSH    H       ;YES, SAVE 1ST <EXPR4>
0680 CDA706             CALL    EXPR4   ;AND GET 2ND ONE
0683 0600               MVI     B,0Q    ;CLEAR B FOR SIGN
0685 CDA307             CALL    CHKSGN  ;CHECK SIGN OF 2ND
0688 EB                 XCHG            ;PUT 2ND IN DE
0689 E3                 XTHL            ;GET 1ST IN HL
068A CDA307             CALL    CHKSGN  ;CHECK SIGN OF 1ST
068D 7A                 MOV     A,D     ;DIVIDE BY 0?
068E B3                 ORA     E
068F CA6701             JZ      AHOW    ;SAY "HOW?"
0692 C5                 PUSH    B       ;ELSE SAVE SIGN
0693 CD8607             CALL    DIVIDE  ;USE SUBROUTINE
0696 60                 MOV     H,B     ;RESULT IN HL NOW
0697 69                 MOV     L,C
0698 C1                 POP     B       ;GET SIGN BACK
0699 D1         XP35    POP     D       ;AND TEXT POINTER
069A 7C                 MOV     A,H     ;HL MUST BE +
069B B7                 ORA     A
069C FA6601             JM      QHOW    ;ELSE IT IS OVERFLOW
069F 78                 MOV     A,B
06A0 B7                 ORA     A
06A1 FCA607             CM      CHGSGN  ;CHANGE SIGN IFF NEEDED
06A4 C34D06             JMP     XP31    ;LOOK OR MORE TERMS
                ;*
06A7 217E02     EXPR4   LXI     H,TAB4-1  ;FIND FUNCTION IN TAB4
06AA C3D802             JMP     EXEC    ;AND GO DO IT
06AD FF         XP40    RST     7       ;NO, NOT A FUNCTION
06AE DAB606             JC      XP41    ;NOR A VARIABLE
06B1 7E                 MOV     A,M     ;VARIABLE
06B2 23                 INX     H
06B3 66                 MOV     H,M     ;VALUE IN HL
06B4 6F                 MOV     L,A
06B5 C9                 RET
06B6 CD3E01     XP41    CALL    TSTNUM  ;OR IS IT A NUMBER
06B9 78                 MOV     A,B     ;# OF DIGIT
06BA B7                 ORA     A
06BB C0                 RNZ             ;OK
06BC CF         PARN    RST     1       ;NO DIGIT, MUST BE
06BD 28                 DB      '('
06BE 05                 DB      5Q
06BF DF                 RST     3        ;"(EXPR)"
06C0 CF                 RST     1
06C1 29                 DB      ')'
06C2 01                 DB      1Q
06C3 C9         XP42    RET
06C4 C3E007     XP43    JMP     QWHAT   ;ELSE SAY: "WHAT?"
                ;*
06C7 CDBC06     RND     CALL    PARN    ;*** RND(EXPR) ***
06CA 7C                 MOV     A,H     ;EXPR MUST BE +
06CB B7                 ORA     A
06CC FA6601             JM      QHOW
06CF B5                 ORA     L       ;AND NON-ZERO
06D0 CA6601             JZ      QHOW
06D3 D5                 PUSH    D       ;SAVE BOTH
06D4 E5                 PUSH    H
06D5 2AD309             LHLD    RANPNT  ;GET MEMORY AS RANDOM
06D8 11B009             LXI     D,LSTROM        ;NUMBER
06DB E7                 RST     4
06DC DAE206             JC      RA1     ;WRAP AROUND IFF LAST
06DF 210001             LXI     H,START
06E2 5E         RA1     MOV     E,M
06E3 23                 INX     H
06E4 56                 MOV     D,M
06E5 22D309             SHLD    RANPNT
06E8 E1                 POP     H
06E9 EB                 XCHG
06EA C5                 PUSH    B
06EB CD8607             CALL    DIVIDE  ;RND(N)=MOD(M,N)+1
06EE C1                 POP     B
06EF D1                 POP     D
06F0 23                 INX     H
06F1 C9                 RET
                ;*
06F2 CDBC06     ABS     CALL    PARN    ;*** ABS(EXPR) ***
06F5 CDA307             CALL    CHKSGN  ;CHECK SIGN
06F8 7C                 MOV     A,H     ;NOTE THAT -32768
06F9 B4                 ORA     H       ;CANNOT CHANGE SIGN
06FA FA6601             JM      QHOW    ;SO SAY: "HOW?"
06FD C9                 RET
```

```
06FE 2AD509     SIZE    LHLD    TXTUNF  ;*** SIZE ***
0701 D5                 PUSH    D       ;GET THE NUMBER OF FREE
0702 EB                 XCHG            ;BYTES BETWEEN 'TXTUNF'
0703 21000F     SIZEA   LXI     H,VARBGN        ;AND 'VARBGN'
0706 CD9C07             CALL    SUBDE
0709 D1                 POP     D
070A C9                 RET
                ;*
                ;*************************************************************
                ;*
                ;*      *** OUT *** INP *** WAIT *** POKE *** PEEK *** & USR
                ;*
                ;*      OUT I,J(,K,L)
                ;*
                ;*      OUTPUTS EXPRESSION 'J' TO PORT 'I', AND MAY BE REPEATED
                ;*      AS IN DATA 'L' TO PORT 'K' AS MANY TIMES AS NEEDED
                ;*      THIS COMMAND MODIFIES ;*        THIS COMMAND MODIFIES
                ;*      THIS COMMAND MODIFY'S A SMALL SECTION OF CODE LOCATED
                ;*      JUST ABOVE ADDRESS 2K
                ;*
                ;*      INP (I)
                ;*
                ;*      THIS FUNCTION RETURNS DATA READ FROM INPUT PORT 'I' AS
                ;*      IT'S VALUE.
                ;*      IT ALSO MODIFIES CODE JUST ABOVE 2K.
                ;*
                ;*      WAIT I,J,K
                ;*
                ;*      THIS COMMAND READS THE STATUS OF PORT 'I', EXCLUSIVE OR'S
                ;*      THE RESULT WITH 'K' IF THERE IS ONE, OR IF NOT WITH 0,
                ;*      AND'S WITH 'J' AND RETURNS WHEN THE RESULT IS NONZERO.
                ;*      ITS MODIFIED CODE IS ALSO ABOVE 2K.
                ;*
                ;*      POKE I,J(,K,L)
                ;*
                ;*      THIS COMMAND WORKS LIKE OUT EXCEPT THAT IT PUTS DATA 'J'
                ;*      INTO MEMORY LOCATION 'I'.
                ;*
                ;*      PEEK (I)
                ;*
                ;*      THIS FUNCTION WORKS LIKE INP EXCEPT IT GETS IT'S VALUE
                ;*      FROM MEMORY LOCATION 'I'.
                ;*
                ;*      USR (I(,J))
                ;*
                ;*      USR CALLS A MACHINE LANGUAGE SUBROUTINE AT LOCATION 'I'
                ;*      IF THE OPTIONAL PARAMETER 'J' IS USED ITS VALUE IS PASSED
                ;*      IN H&L. THE VALUE OF THE FUNCTION SHOULD BE RETURNED IN H&
                ;*
                ;*************************************************************
                ;*
070B DF         OUTCMD  RST     3
070C 7D                 MOV     A,L
070D 32B109             STA     OUTIO + 1
0710 CF                 RST     1
0711 2C                 DB      ','
0712 2F                 DB      2FH
0713 DF                 RST     3
0714 7D                 MOV     A,L
0715 CDB009             CALL    OUTIO
0718 CF                 RST     1
0719 2C                 DB      ','
071A 03                 DB      03H
071B C30B07             JMP     OUTCMD
071E F7                 RST     6
071F DF         WAITCM  RST     3
0720 7D                 MOV     A,L
0721 32B409             STA     WAITIO + 1
0724 CF                 RST     1
0725 2C                 DB      ','
0726 1B                 DB      1BH
0727 DF                 RST     3
0728 E5                 PUSH    H
0729 CF                 RST     1
072A 2C                 DB      ','
072B 07                 DB      7H
072C DF                 RST     3
072D 7D                 MOV     A,L
072E E1                 POP     H
072F 67                 MOV     H,A
0730 C33207             JMP     $ + 2
0733 2600               MVI     H,0
```

```
0735 C3B309             JMP     WAITIO
0738 CDBC06     INP     CALL    PARN
073B 7D                 MOV     A,L
073C 32BC09             STA     INPIO + 1
073F 2600               MVI     H,0
0741 C3BB09             JMP     INPIO
0744 C3E007             JMP     QWHAT
0747 DF         POKE    RST     3
0748 E5                 PUSH    H
0749 CF                 RST     1
074A 2C                 DB      ','
074B 12                 DB      12H
074C DF                 RST     3
074D 7D                 MOV     A,L
074E E1                 POP     H
074F 77                 MOV     M,A
0750 CF                 RST     1
0751 2C03               DB      ',',03H
0753 C34707             JMP     POKE
0756 F7                 RST 6
0757 CDBC06     PEEK    CALL    PARN
075A 6E                 MOV     L,M
075B 2600               MVI     H,0
075D C9                 RET
075E C3E007             JMP     QWHAT
0761 C5         USR     PUSH    B
0762 CF                 RST     1
0763 281C               DB      '(',28D ;QWHAT
0765 DF                 RST     3       ;EXPR
0766 CF                 RST     1
0767 2907               DB      ')',7   ;PASPARM
0769 D5                 PUSH    D
076A 118007             LXI     D,USRET
076D D5                 PUSH    D
076E E5                 PUSH    H
076F C9                 RET             ;CALL   USR ROUTINE
0770 CF         PASPRM  RST     1
0771 2C0E               DB      ',',14D
0773 E5                 PUSH    H
0774 DF                 RST     3
0775 CF                 RST     1
0776 2909               DB      ')',9
0778 C1                 POP     B
0779 D5                 PUSH    D
077A 118007             LXI     D,USRET
077D D5                 PUSH    D
077E C5                 PUSH    B
077F C9                 RET             ;CALL   USR ROUTINE
0780 D1         USRET   POP     D
0781 C1                 POP     B
0782 C9                 RET
0783 C3E007             JMP     QWHAT
                ;*
                ;*****************************************************************
                ;*
                ;* *** DIVIDE *** SUBDE *** CHKSGN *** CHGSGN *** & CKHLDE ***
                ;*
                ;* 'DIVIDE' DIVIDES HL BY DE, RESULT IN BC, REMAINDER IN HL
                ;*
                ;* 'SUBDE' SUBTRACTS DE FROM HL
                ;*
                ;* 'CHKSGN' CHECKS SIGN OF HL.  IFF +, NO CHANGE. IFF -, CHANGE
                ;* SIGN AND FLIP SIGN OF B.
                ;*
                ;* 'CHGSGN' CHNGES SIGN OF HL AND B UNCONDITIONALLY.
                ;*
                ;* 'CKHLE' CHECKS SIGN OF HL AND DE.    IFF DIFFERENT, HL AND DE
                ;* ARE INTERCHANGED.    IFF SAME SIGN, NOT INTERCHANGED.  EITHER
                ;* CASE, HL DE ARE THEN COMPARED TO SET THE FLAGS.
                ;*
0786 E5         DIVIDE PUSH     H       ;*** DIVIDE ***
0787 6C                MOV      L,H     ;DIVIDE H BY DE
0788 2600              MVI      H,0
078A CD9107            CALL     DV1
078D 41                MOV      B,C     ;SAVE RESULT IN B
078E 7D                MOV      A,L     ;(REMAINDER+L)/DE
078F E1                POP      H
0790 67                MOV      H,A
0791 0EFF       DV1    MVI      C,377Q  ;RESULT IN C
0793 0C         DV2    INR      C       ;DUMB ROUTINE
0794 CD9C07            CALL     SUBDE   ;DIVIDE BY SUBTRACT
0797 D29307            JNC      DV2     ;AND COUNT
```

```
079A 19                 DAD     D
079B C9                 RET
                ;*
079C 7D         SUBDE   MOV     A,L     ;*** SUBDE ***
079D 93                 SUB     E       ;SUBTRACT DE FROM
079E 6F                 MOV     L,A     ;HL
079F 7C                 MOV     A,H
07A0 9A                 SBB     D
07A1 67                 MOV     H,A
07A2 C9                 RET
                ;*
07A3 7C         CHKSGN  MOV     A,H     ;*** CHKSGN ***
07A4 B7                 ORA     A       ;CHECK SIGN OF HL
07A5 F0                 RP              ;IFF -, CHANGE SIGN
                ;*
07A6 7C         CHGSGN  MOV     A,H     ;*** CHGSGN ***
07A7 2F                 CMA             ;CHANGE SIGN OF HL
07A8 67                 MOV     H,A
07A9 7D                 MOV     A,L
07AA 2F                 CMA
07AB 6F                 MOV     L,A
07AC 23                 INX     H
07AD 78                 MOV     A,B     ;AND ALSO FLIP B
07AE EE80               XRI     200Q
07B0 47                 MOV     B,A
07B1 C9                 RET
                ;*
07B2 7C         CKHLDE  MOV     A,H
07B3 AA                 XRA     D       ;SAME SIGN?
07B4 F2B807             JP      CK1     ;YES, COMPARE
07B7 EB                 XCHG            ;NO, XCH AND COMP
07B8 E7         CK1     RST     4
07B9 C9                 RET
                ;*
                ;***************************************************************
                ;*
                ;* *** SETVAL *** FIN *** ENDCHK *** & ERROR (& FRIENDS) ***
                ;*
                ;* "SETVAL" EXPECTS A VARIABLE, FOLLOWED BY AN EQUAL SIGN AND
                ;* THEN AN EXPR. IT EVALUATES THE EXPR. AND SET THE VARIABLE
                ;* TO THAT VALUE.
                ;*
                ;* "FIN" CHECKS THE END OF A COMMAND. IFF IT ENDED WITH ";",
                ;* EXECUTION CONTINUES. IFF IT ENDED WITH A CR, IT FINDS THE
                ;* NEXT LINE AND CONTINUE FROM THERE.
                ;*
                ;* "ENDCHK" CHECKS IFF A COMMAND IS ENDED WITH CR. THIS IS
                ;* REQUIRED IN CERTAIN COMMANDS. (GOTO, RETURN, AND STOP ETC.)
                ;*
                ;* "ERROR" PRINTS THE STRING POINTED BY DE (AND ENDS WITH CR).
                ;* IT THEN PRINTS THE LINE POINTED BY 'CURRNT' WITH A "?"
                ;* INSERTED AT WHERE THE OLD TEXT POINTER (SHOULD BE ON TOP
                ;* O THE STACK) POINTS TO. EXECUTION OF TB IS STOPPED
                ;* AND TBI IS RESTARTED. HOWEVER, IFF 'CURRNT' -> ZERO
                ;* (INDICATING A DIRECT COMMAND)), THE DIRECT COMMAND IS NOT
                ;* PRINTED, AND IFF 'CURRNT' -> NEGATIVE # (INDICATING 'INPUT'
                ;* COMMAND, THE INPUT LINE IS NOT PRINTED AND EXECUTION IS
                ;* NOT TERMINATED BUT CONTINUED AT 'INPERR'.
                ;*
                ;* RELATED TO 'ERROR' ARE THE FOLLOWING:
                ;* 'QWHAT' SAVES TEXT POINTER IN STACK AND GET MESSAGE "WHAT?"
                ;* 'AWHAT' JUST GET MESSAGE "WHAT?" AND JUMP TO 'ERROR'.
                ;* 'QSORRY' AND 'ASORRY' DO SAME KIND OF THING.
                ;* 'QHOW' AND 'AHOW' IN THE ZERO PAGE SECTION ALSO DO THIS
                ;*
07BA FF         SETVAL  RST     7       ;*** SETVAL ***
07BB DAE007             JC      QWHAT   ;"WHAT?" NO VARIABLE
07BE E5                 PUSH    H       ;SAVE ADDRESS OF VAR.
07BF CF                 RST     1       ;PASS "=" SIGN
07C0 3D                 DB      '='
07C1 08                 DB      10Q
07C2 DF                 RST     3       ;EVALUATE EXPR.
07C3 44                 MOV     B,H     ;VALUE IN BC NOW
07C4 4D                 MOV     C,L
07C5 E1                 POP     H       ;GET ADDRESS
07C6 71                 MOV     M,C     ;SAVE VALUE
07C7 23                 INX     H
07C8 70                 MOV     M,B
07C9 C9                 RET
07CA C3E007     SV1     JMP     QWHAT   ;NO "=" SIGN
                ;*
```

```
07CD CF         FIN     RST     1       ;*** FIN ***
07CE 3B                 DB      73Q
07CF 04                 DB      4Q
07D0 F1                 POP     PSW     ;";", PURGE RET ADDR.
07D1 C32B03             JMP     RUNSML  ;CONTINUE SAME LINE
07D4 CF         FI1     RST     1       ;NOT ";", IS IT CR?
07D5 0D                 DB      0DH
07D6 04                 DB      4Q
07D7 F1                 POP     PSW     ;YES, PURGE RET ADDR.
07D8 C31B03             JMP     RUNNXL  ;RUN NEXT LINE
07DB C9         FI2     RET             ;ELSE RETURN TO CALLER
                ;*
07DC EF         ENDCHK  RST     5       ;*** ENDCHK ***
07DD FE0D               CPI     0DH     ;END WITH CR?
07DF C8                 RZ              ;OK, ELSE SAY: "WHAT?"
                ;*
07E0 D5         QWHAT   PUSH    D       ;*** QWHAT ***
07E1 117501     AWHAT   LXI     D,WHAT  ;*** AWHAT ***
07E4 97         ERROR   SUB     A       ;*** ERROR ***
07E5 CD7F08             CALL    PRTSTG  ;PRINT 'WHAT?', 'HOW?'
07E8 D1                 POP     D       ;OR 'SORRY'
07E9 1A                 LDAX    D       ;SAVE THE CHARACTER
07EA F5                 PUSH    PSW     ;AT WHERE OLD DE ->
07EB 97                 SUB     A       ;AND PUT A 0 THERE
07EC 12                 STAX    D
07ED 2AC109             LHLD    CURRNT  ;GET CURRENT LINE #
07F0 E5                 PUSH    H
07F1 7E                 MOV     A,M     ;CHECK THE VALUE
07F2 23                 INX     H
07F3 B6                 ORA     M
07F4 D1                 POP     D
07F5 CA8101             JZ      RSTART  ;IFF ZERO, JUST RERSTART
07F8 7E                 MOV     A,M     ;IFF NEGATIVE,
07F9 B7                 ORA     A
07FA FA6805             JM      INPERR  ;REDO INPUT
07FD CDED08             CALL    PRTLN   ;ELSE PRINT THE LINE
0800 1B                 DCX     D       ;UPTO WHERE THE 0 IS
0801 F1                 POP     PSW     ;RESTORE THE CHARACTER
0802 12                 STAX    D
0803 3E3F               MVI     A,77Q   ;PRINTT A "?"
0805 D7                 RST     2
0806 97                 SUB     A       ;AND THE REST OF THE
0807 CD7F08             CALL    PRTSTG  ;LINE
080A C38101             JMP     RSTART
080D D5         QSORRY  PUSH    D       ;*** QSORRY ***
080E 117B01     ASORRY  LXI     D,SORRY ;*** ASORRY ***
0811 C3E407             JMP     ERROR
                ;*
                ;*****************************************************************
                ;*
                ;* *** GETLN *** FNDLN (& FRIENDS) ***
                ;*
                ;* 'GETLN' READS A INPUT LINE INTO 'BUFFER'. IT FIRST PROMPT
                ;* THE CHARACTER IN A (GIVEN BY THE CALLER), THEN IT FILLS THE
                ;* THE BUFFER AND ECHOS. IT IGNORES LF'S AND NULLS, BUT STILL
                ;* ECHOS THEM BACK. RUB-OUT IS USED TO CAUSE IT TO DELETE
                ;* THE LAST CHARATER (IFF THERE IS ONE), AND ALT-MOD IS USED TO
                ;* CAUSE IT TO DELETE THE WHOLE LINE AND START IT ALL OVER.
                ;* ODHSIGNALS THE END OF A LINE, AND CAUE 'GETLN' TO RETURN.
                ;*
                ;* 'FNDLN' FINDS A LINE WITH A GIVEN LINE # (IN HL) IN THE
                ;* TEXT SAVE AREA. DE IS USED AS THE TEXT POINTER. IFF THE
                ;* LINE IS FOUND, DE WILL POINT TO THE BEGINNING OF THAT LINE
                ;* (I.E., THE LOW BYTE OF THE LINE #), AND FLAGS ARE NC & Z.
                ;* IFF THAT LINE IS NOT THERE AND A LINE WITH A HIGHER LINE #
                ;* IS FOUND, DE POINTS TO THERE AND FLAGS ARE NC & NZ.  IFF
                ;* WE REACHED THE END OF TEXT SAVE ARE AND CANNOT FIND THE
                ;* LINE, FLAGS ARE C & NZ.
                ;* 'FNDLN' WILL INITIALIZE DE TO THE BEGINNING OF THE TEXT SAVE
                ;* AREA TO START THE SEARCH. SOME OTHER ENTRIES OF THIS
                ;* ROUTINE WILL NOT INITIALIZE DE AND DO THE SEARCH.
                ;* 'FNDLNP' WILL START WITH DE AND SEARCH FOR THE LINE #.
                ;* 'FNDNXT' WILL BUMP DE BY 2, FIND A 0DHAND THEN START SEARCH.
                ;* 'FNDSKP' USE DE TO FIND A CR, AND THEN STRART SEARCH.
                ;*
0814 D7         GETLN   RST     2       ;*** GETLN ***
0815 11370F             LXI     D,BUFFER  ;PROMPT AND INIT
0818 CD8509     GL1     CALL    CHKIO   ;CHECK KEYBOARD
081B CA1808             JZ      GL1     ;NO INPUT, WAIT
081E FE7F               CPI     177Q    ;DELETE LST CHARACTER?
0820 CA4208             JZ      GL3     ;YES
0823 FE0A               CPI     12Q     ;IGNORE LF
```

```
0825 CA1808            JZ      GL1
0828 B7                ORA     A       ;IGNORE NULL
0829 CA1808            JZ      GL1
082C FE5C              CPI     134Q    ;DELETE THE WHOLE LINE?
082E CA4F08            JZ      GL4     ;YES
0831 12                STAX    D       ;ELSE, SAVE INPUT
0832 13                INX     D       ;AND BUMP POINTER
0833 FE0D              CPI     15Q     ;WAS IT CR?
0835 C23C08            JNZ     GL2     ;NO
0838 3E0A              MVI     A,12Q   ;YES, GET LINE FEED
083A D7                RST     2       ;CALL   OUTC AND LINE FEED
083B C9                RET             ;WE'VE GOT A LINE
083C 7B        GL2     MOV     A,E     ;MORE FREE ROOM?
083D FE87              CPI     BUFEND AND 0FFH
083F C21808            JNZ     GL1     ;YES, GET NEXT INPUT
0842 7B        GL3     MOV     A,E     ;DELETE LAST CHARACTER
0843 FE37              CPI     BUFFER AND 0FFH ;BUT DO WE HAVE ANY?
0845 CA4F08            JZ      GL4     ;NO, REDO WHOLE LINE
0848 1B                DCX     D       ;YES, BACKUP POINTER
0849 3E5F              MVI     A,'_'   ;AND ECHO A BACK-SPACE
084B D7                RST     2
084C C31808            JMP     GL1     ;GO GET NEXT INPUT
084F CD0E00    GL4     CALL    CRLF    ;REDO ENTIRE LINE
0852 3E5E              MVI     A,136Q  ;CR, LF AND UP-ARROW
0854 C31408            JMP     GETLN
               ;*
0857 7C        FNDLN   MOV     A,H     ;*** FNDLN ***
0858 B7                ORA     A       ;CHECK SIGN OF HL
0859 FA6601            JM      QHOW    ;IT CANNT BE -
085C 11D709            LXI     D,TXTBGN  ;INIT. TEXT POINTER
               ;*
085F =         FNDLNP  EQU     $       ;*** FNDLNP ***
085F E5        FL1     PUSH    H       ;SAVE LINE #
0860 2AD509            LHLD    TXTUNF  ;CHECK IFF WE PASSED END
0863 2B                DCX     H
0864 E7                RST     4
0865 E1                POP     H       ;GET LINE # BACK
0866 D8                RC              ;C,NZ PASSED END
0867 1A                LDAX    D       ;WE DID NOT, GET BYTE 1
0868 95                SUB     L       ;IS THIS THE LINE?
0869 47                MOV     B,A     ;COMPARE LOW ORDER
086A 13                INX     D
086B 1A                LDAX    D       ;GET BYTE 2
086C 9C                SBB     H       ;COMPARE HIGH ORDER
086D DA7408            JC      FL2     ;NO, NOT THERE YET
0870 1B                DCX     D       ;ELSE WE EITHER FOUND
0871 B0                ORA     B       ;IT, OR IT IS NOT THERE
0872 C9                RET             ;NC,Z:FOUND; NC,NZ:NO
               ;*
0873 =         FNDNXT  EQU     $       ;*** FNDNXT ***
0873 13                INX     D       ;FIND NEXT LINE
0874 13        FL2     INX     D       ;JUST PASSED BYTE 1 & 2
               ;*
0875 1A        FNDSKP  LDAX    D       ;*** FNDSKP ***
0876 FE0D              CPI     0DH     ;TRY TO FIND 0DH
0878 C27408            JNZ     FL2     ;KEEP LOOKING
087B 13                INX     D       ;FOUND CR, SKIP OVER
087C C35F08            JMP     FL1     ;CHECK IFF END OF TEXT
               ;*
               ;*************************************************************
               ;*
               ;* *** PRTSTG *** QTSTG *** PRTNUM *** & PRTLN ***
               ;*
               ;* 'PRTSTG' PRINTS A STRING POINTED BY DE. IT STOPS PRINTING
               ;* AND RETURNS TO CALLER WHEN EITHER A 0DHIS PRINTED OR WHEN
               ;* THE NEXT BYTE IS THE SAME AS WHAT WAS IN A (GIVEN BY THE
               ;* CALLER). OLD A IS STORED IN B, OLD B IS LOST.
               ;*
               ;* 'QTSTG' LOOKS FOR A BACK-ARROW, SINGLE QUOTE, OR DOUBLE
               ;* QUOTE. IFF NONE OF THESE, RETURN TO CALLER. IFF BACK-ARROW,
               ;* OUTPUT A 0DHWITHOUT A LF.  IFF SINGLE OR DOUBLE QUOTE, PRINT
               ;* THE STRING IN THE QUOTE AND DEMANDS A MATCHING UNQUOTE.
               ;* AFTER THE PRINTING THE NEXT 3 BYTES OF THE CALLER IS SKIPPED
               ;* OVER (USUALLY A JUMP INSTRUCTION).
               ;*
               ;* 'PRTNUM' PRINTS THE NUMBER IN HL. LEADING BLANKS ARE ADDED
               ;* IFF NEEDED TO PAD THE NUMBER OF SPACES TO THE NUMBER IN C.
               ;* HOWEVER, IFF THE NUMBER OF DIGITS IS LARGER THAN THE # IN
               ;* C, ALL DIGITS ARE PRINTED ANYWAY. NEGATIVE SIGN IS ALSO
               ;* PRINTED AND COUNTED IN, POSITIVE SIGN IS NOT.
               ;*
               ;* 'PRTLN' PRINSRA SAVED TEXT LINE WITH LINE # AND ALL.
```

```
                ;*
087F 47         PRTSTG  MOV     B,A     ;*** PRTSTG ***
0880 1A         PS1     LDAX    D       ;GET A CHARACTERR
0881 13                 INX     D       ;BUMP POINTER
0882 B8                 CMP     B       ;SAME AS OLD A?
0883 C8                 RZ              ;YES, RETURN
0884 D7                 RST     2       ;ELSE PRINT IT
0885 FE0D               CPI     0DH     ;WAS IT A CR?
0887 C28008             JNZ     PS1     ;NO, NEXT
088A C9                 RET             ;YES, RETURN
                ;*
088B CF         QTSTG   RST     1       ;*** QTSTG ***
088C 22                 DB      '"'
088D 0F                 DB      17Q
088E 3E22               MVI     A,42Q   ;IT IS A "
0890 CD7F08     QT1     CALL    PRTSTG  ;PRINT UNTIL ANOTHER
0893 FE0D               CPI     0DH     ;WAS LAST ONE A CR?
0895 E1                 POP     H       ;RETURN ADDRESS
0896 CA1B03             JZ      RUNNXL  ;WAS CR, RUN NEXT LINE
0899 23         QT2     INX     H       ;SKIP 3 BYTES ON RETURN
089A 23                 INX     H
089B 23                 INX     H
089C E9                 PCHL            ;RETURN
089D CF         QT3     RST     1       ;IS IT A ' ?
089E 27                 DB      47Q
089F 05                 DB      5Q
08A0 3E27               MVI     A,47Q   ;YES, DO SAME
08A2 C39008             JMP     QT1     ;AS IN "
08A5 CF         QT4     RST     1       ;IS IT BACK-ARROW?
08A6 5F                 DB      137Q
08A7 08                 DB      10Q
08A8 3E8D               MVI     A,215Q  ;YES, ODHWITHOUT LF!!
08AA D7                 RST     2       ;DO IT TWICE TO GIVE
08AB D7                 RST     2       ;TTY ENOUGH TIME
08AC E1                 POP     H       ;RETURN ADDRESS
08AD C39908             JMP     QT2
08B0 C9         QT5     RET             ;NONE OF ABOVE
                ;*
08B1 D5         PRTNUM  PUSH    D       ;*** PRTNUM ***
08B2 110A00             LXI     D,12Q   ;DECIMAL
08B5 D5                 PUSH    D       ;SAVE AS A FLAG
08B6 42                 MOV     B,D     ;B=SIGN
08B7 0D                 DCR     C       ;C=SPACES
08B8 CDA307             CALL    CHKSGN  ;CHECK SIGN
08BB F2C108             JP      PN1     ;NO SIGN
08BE 062D               MVI     B,55Q   ;B=SIGN
08C0 0D                 DCR     C       ;'-' TAKES SPACE
08C1 C5         PN1     PUSH    B       ;SAVE SIGN & SPACE
08C2 CD8607     PN2     CALL    DIVIDE  ;DEVIDE HL BY 10
08C5 78                 MOV     A,B     ;RESULT 0?
08C6 B1                 ORA     C
08C7 CAD208             JZ      PN3     ;YES, WE GOT ALL
08CA E3                 XTHL            ;NO, SAVE REMAINDER
08CB 2D                 DCR     L       ;AND COUNT SPACE
08CC E5                 PUSH    H       ;HL IS OLD BC
08CD 60                 MOV     H,B     ;MOVE RESULT TO BC
08CE 69                 MOV     L,C
08CF C3C208             JMP     PN2     ;AND DIVIDE BY 10
08D2 C1         PN3     POP     B       ;WE GOT ALL DIGITS IN
08D3 0D         PN4     DCR     C       ;THE STACK
08D4 79                 MOV     A,C     ;LOOK AT SPACE COUNT
08D5 B7                 ORA     A
08D6 FADF08             JM      PN5     ;NO LEADING BLANKS
08D9 3E20               MVI     A,40Q   ;LEADING BLANKS
08DB D7                 RST     2
08DC C3D308             JMP     PN4     ;MORE?
08DF 78         PN5     MOV     A,B     ;PRINT SIGN
08E0 D7                 RST     2       ;MAYBE - OR NULL
08E1 5D                 MOV     E,L     ;LAST REMAINDER IN E
08E2 7B         PN6     MOV     A,E     ;CHECK DIGIT IN E
08E3 FE0A               CPI     12Q     ;10 IS FLAG FOR NO MORE
08E5 D1                 POP     D
08E6 C8                 RZ              ;IFF SO, RETURN
08E7 C630               ADI     60Q     ;ELSE CONVERT TO ASCII
08E9 D7                 RST     2       ;AND PRINT THE DIGIT
08EA C3E208             JMP     PN6     ;GO BACK FOR MORE
                ;*
08ED 1A         PRTLN   LDAX    D       ;*** PRTLN ***
08EE 6F                 MOV     L,A     ;LOW ORDER LINE #
08EF 13                 INX     D
08F0 1A                 LDAX    D       ;HIGH ORDER
08F1 67                 MOV     H,A
```

```
08F2 13             INX     D
08F3 0E04           MVI     C,4Q    ;PRINT 4 DIGIT LINE #
08F5 CDB108         CALL    PRTNUM
08F8 3E20           MVI     A,40Q   ;FOLLOWED BY A BLANK
08FA D7             RST     2
08FB 97             SUB     A       ;AND THEN THE TEXT
08FC CD7F08         CALL    PRTSTG
08FF C9             RET
                ;*
                ;*****************************************************************
                ;*
                ;* *** MVUP *** MVDOWN *** POPA *** & PUSHA ***
                ;*
                ;* 'MVUP' MOVES A BLOCK UP FROM HERE DE-> TO WHERE BC-> UNTIL
                ;* DE = HL
                ;*
                ;* 'MVDOWN' MOVES A BLOCK DOWN FROM WHERE DE-> TO WHERE HL->
                ;* UNTIL DE = BC
                ;*
                ;* 'POPA' RESTORES THE 'FOR' LOOP VARIABLE SAVE AREA FROM THE
                ;* STACK
                ;*
                ;* 'PUSHA' STACKS THE 'FOR' LOOP VARIABLE SAVE AREA INTO THE
                ;* STACK
                ;*
0900 E7         MVUP    RST     4       ;*** MVUP ***
0901 C8                 RZ              ;DE = HL, RETURN
0902 1A                 LDAX    D       ;GET ONE BYTE
0903 02                 STAX    B       ;MOVE IT
0904 13                 INX     D       ;INCREASE BOTH POINTERS
0905 03                 INX     B
0906 C30009             JMP     MVUP    ;UNTIL DONE
                ;*
0909 78         MVDOWN  MOV     A,B     ;*** MVDOWN ***
090A 92                 SUB     D       ;TEST IFF DE = BC
090B C21109             JNZ     MD1     ;NO, GO MOVE
090E 79                 MOV     A,C     ;MAYBE, OTHER BYTE?
090F 93                 SUB     E
0910 C8                 RZ              ;YES, RETURN
0911 1B         MD1     DCX     D       ;ELSE MOVE A BYTE
0912 2B                 DCX     H       ;BUT FIRST DECREASE
0913 1A                 LDAX    D       ;BOTH POINTERS AND
0914 77                 MOV     M,A     ;THEN DO IT
0915 C30909             JMP     MVDOWN  ;LOOP BACK
                ;*
0918 C1         POPA    POP     B       ;BC = RETURN ADDR.
0919 E1                 POP     H       ;RESTORE LOPVAR, BUT
091A 22C909             SHLD    LOPVAR  ;=0 MEANS NO MORE
091D 7C                 MOV     A,H
091E B5                 ORA     L
091F CA3209             JZ      PP1     ;YEP, GO RETURN
0922 E1                 POP     H       ;NOP, RESTORE OTHERS
0923 22CB09             SHLD    LOPINC
0926 E1                 POP     H
0927 22CD09             SHLD    LOPLMT
092A E1                 POP     H
092B 22CF09             SHLD    LOPLN
092E E1                 POP     H
092F 22D109             SHLD    LOPPT
0932 C5         PP1     PUSH    B       ;BC = RETURN ADDR.
0933 C9                 RET
                ;*
0934 21AF0F     PUSHA   LXI     H,STKLMT  ;*** PUSHA ***
0937 CDA607             CALL    CHGSGN
093A C1                 POP     B       ;BC=RETURN ADDRESS
093B 39                 DAD     SP      ;IS STACK NEAR THE TOP?
093C D20D08             JNC     QSORRY  ;YES, SORRY FOR THAT.
093F 2AC909             LHLD    LOPVAR  ;ELSE SAVE LOOP VAR.S
0942 7C                 MOV     A,H     ;BUT IFF LOPVAR IS 0
0943 B5                 ORA     L       ;THAT WILL BE ALL
0944 CA5A09             JZ      PU1
0947 2AD109             LHLD    LOPPT   ;ELSE, MORE TO SAVE
094A E5                 PUSH    H
094B 2ACF09             LHLD    LOPLN
094E E5                 PUSH    H
094F 2ACD09             LHLD    LOPLMT
0952 E5                 PUSH    H
0953 2ACB09             LHLD    LOPINC
0956 E5                 PUSH    H
0957 2AC909             LHLD    LOPVAR
095A E5         PU1     PUSH    H
095B C5                 PUSH    B       ;BC = RETURN ADDR.
```

```
095C C9                 RET
                        ;*
                        ;*****************************************************************
                        ;*
                        ;* *** OUTC *** & CHKIO ***
                        *!
                        ;* THESE ARE THE ONLY I/O ROUTINES IN TBI.
                        ;* 'OUTC' IS CONTROLLED BY A SOFTWARE SWITCH 'OCSW'. IFF OCSW=0
                        ;* 'OUTC' WILL JUST RETURN TO THE CALLER. IFF OCSW IS NOT 0,
                        ;* IT WILL OUTPUT THE BYTE IN A. IFF THAT IS A CR, A LF IS ALSO
                        ;* SEND OUT.    ONLY THE FLAGS MAY BE CHANGED AT RETURN, ALL REG.
                        ;* ARE RESTORED.
                        ;*
                        ;* 'CHKIO' CHECKS THE INPUT. IFF NO INPUT, IT WILL RETURN TO
                        ;* THE CALLER WITH THE Z FLAG SET. IFF THERE IS INPUT, Z FLAG
                        ;* IS CLEARED AND THE INPUT BYTE IS IN A. HOWERER, IFF THE
                        ;* INPUT IS A CONTROL-O, THE 'OCSW' SWITCH IS COMPLIMENTED, AND
                        ;* Z FLAG IS RETURNED.  IFF A CONTROL-C IS READ, 'CHKIO' WILL
                        ;* RESTART TBI AND DO NOT RETURN TO THE CALLER.
                        ;*
095D C26209     OC2     JNZ     OC3     ;IT IS ON
0960 F1                 POP     PSW     ;IT IS OFF
0961 C9                 RET             ;RESTORE AF AND RETURN
0962 F1         OC3     POP     A       ;GET OLD A BACK
0963 C5                 PUSH    B       ;SAVE B ON STACK
0964 D5                 PUSH    D       ;AND D
0965 E5                 PUSH    H       ;AND H TOO
0966 32BF09             STA     OUTCAR  ;SAVE CHARACTER
0969 5F                 MOV     E,A     ;PUT CHAR. IN E FOR CPM
096A 0E02               MVI     C,2     ;GET CONOUT COMMAND
096C CD0500             CALL    CPM     ;CALL   CPM AND DO IT
096F 3ABF09             LDA     OUTCAR  ;GET CHAR. BACK
0972 FE0D               CPI     0DH     ;WAS IT A 'CR'?
0974 C27E09             JNZ     DONE    ;NO, DONE
0977 1E0A               MVI     E,0AH   ;GET LINEFEED
0979 0E02               MVI     C,2     ;AND CONOUT AGAIN
097B CD0500             CALL    CPM     ;CALL   CPM
097E 3ABF09     DONE    LDA     OUTCAR  ;GET CHARACTER BACK
0981 E1         IDONE   POP     H       ;GET H BACK
0982 D1                 POP     D       ;AND D
0983 C1                 POP     B       ;AND B TOO
0984 C9                 RET             ;DONE AT LAST
0985 C5         CHKIO   PUSH    B       ;SAVE B ON STACK
0986 D5                 PUSH    D       ;AND D
0987 E5                 PUSH    H       ;THEN H
0988 0E0B               MVI     C,11    ;GET CONSTAT WORD
098A CD0500             CALL    CPM     ;CALL   THE BDOS
098D B7                 ORA     A       ;SET FLAGS
098E C29409             JNZ     CI1     ;IF READY GET CHARACTER
0991 C38109             JMP     IDONE   ;RESTORE AND RETURN
0994 0E01       CI1     MVI     C,1     ;GET CONIN WORD
0996 CD0500             CALL    CPM     ;CALL   THE BDOS
0999 FE0F               CPI     0FH     ;IS IT CONTROL-O?
099B C2A809             JNZ     CI2     ;NO, MORE CHECKING
099E 3AC009             LDA     OCSW    ;CONTROL-O      FLIP OCSW
09A1 2F                 CMA             ;ON TO OFF, OFF TO ON
09A2 32C009             STA     OCSW    ;AND PUT IT BACK
09A5 C38509             JMP     CHKIO   ;AND GET ANOTHER CHARACTER
09A8 FE03       CI2     CPI     3       ;IS IT CONTROL-C?
09AA C28109             JNZ     IDONE   ;RETURN AND RESTORE IF NOT
09AD C38101             JMP     RSTART  ;YES, RESTART TBI
09B0 =          LSTROM  EQU     $       ;ALL ABOVE CAN BE ROM
09B0 D3FF       OUTIO   OUT     0FFH
09B2 C9                 RET
09B3 DBFF       WAITIO  IN      0FFH
09B5 AC                 XRA     H
09B6 A5                 ANA     L
09B7 CAB309             JZ      WAITIO
09BA F7                 RST     6
09BB DBFF       INPIO   IN      0FFH
09BD 6F                 MOV     L,A
09BE C9                 RET
09BF 00         OUTCAR  DB      0       ;OUTPUT CHAR. STORAGE
09C0 FF         OCSW    DB      0FFH    ;SWITCH FOR OUTPUT
09C1 0000       CURRNT  DW      0       ;POINTS TO CURRENT LINE
09C3 0000       STKGOS  DW      0       ;SAVES SP IN 'GOSUB'
09C5 0000       VARNXT  DW      0       ;TEMPORARY STORAGE
09C7 0000       STKINP  DW      0       ;SAVES SP IN 'INPUT'
09C9 0000       LOPVAR  DW      0       ;'FOR' LOOP SAVE AREA
```

```
09CB 0000       LOPINC  DW      0       ;INCREMENT
09CD 0000       LOPLMT  DW      0       ;LIMIT
09CF 0000       LOPLN   DW      0       ;LINE NUMBER
09D1 0000       LOPPT   DW      0       ;TEXT POINTER
09D3 0001       RANPNT  DW      START   ;RANDOM NUMBER POINTER
09D5 D709       TXTUNF  DW      TXTBGN  ;->UNFILLED TEXT AREA
09D7            TXTBGN  DS      1       ;TEXT SAVE AREA BEGINS
09D8 7F7F7F5348MSG1     DB      7FH,7FH,7FH,'SHERRY BROTHERS TINY BASIC VER. 3.1',(
09FF 3EFF       INIT    MVI     A,0FFH
0A01 32C009             STA     OCSW    ;TURN ON OUTPUT SWITCH
0A04 3E0C               MVI     A,0CH   ;GET FORM FEED
0A06 D7                 RST     2       ;SEND TO CRT
0A07 97         PATLOP  SUB     A       ;CLEAR ACCUMULATOR
0A08 11D809             LXI     D,MSG1  ;GET INIT MESSAGE
0A0B CD7F08             CALL    PRTSTG  ;SEND IT
0A0E 3A0700     LSTRAM  LDA     7       ;GET FBASE FOR TOP
0A11 328301             STA     RSTART+2
0A14 3D                 DCR     A       ;DECREMENT FOR OTHER POINTERS
0A15 321901             STA     SS1A+2  ;AND FIX THEM TOO
0A18 322601             STA     TV1A+2
0A1B 32A501             STA     ST3A+2
0A1E 32E501             STA     ST4A+2
0A21 32AB05             STA     IP3A+2
0A24 320507             STA     SIZEA+2
0A27 321708             STA     GETLN+3
0A2A 323609             STA     PUSHA+2
0A2D 218401             LXI     H,ST1   ;GET NEW START JUMP
0A30 220101             SHLD    START+1 ;AND FIX IT
0A33 C38401             JMP     ST1
                ;       RESTART TABLE
0A50                    ORG     0A50H
                RSTBL:
0A50 E3                 XTHL            ;*** TSTC OR RST 1 ***
0A51 EF                 RST     5       ;IGNORE BLANKS AND
0A52 BE                 CMP     M       ;TEST CHARACTER
0A53 C32F01             JMP     TC1     ;REST OF THIS IS AT TC1
                ;*
000E =          CRLF:   EQU     0EH     ;EXECUTE TIME LOCATION OF THIS INSTRUCTION
0A56 3E0D               MVI     A,0DH   ;*** CRLF ***
                ;*
0A58 F5                 PUSH    PSW     ;*** OUTC OR RST 2 ***
0A59 3AC009             LDA     OCSW    ;PRINT CHARACTER ONLY
0A5C B7                 ORA     A       ;IFF OCSW SWITCH IS ON
0A5D C35D09             JMP     OC2     ;REST OF THIS IS AT OC2
                ;*
0A60 CD1606             CALL    EXPR2   ;*** EXPR OR RST 3 ***
0A63 E5                 PUSH    H       ;EVALUATE AN EXPRESION
0A64 C3D205             JMP     EXPR1   ;REST OF IT IS AT EXPR1
0A67 57                 DB      'W'
                ;*
0A68 7C                 MOV     A,H     ;*** COMP OR RST 4 ***
0A69 BA                 CMP     D       ;COMPARE HL WITH DE
0A6A C0                 RNZ             ;RETURN CORRECT C AND
0A6B 7D                 MOV     A,L     ;Z FLAGS
0A6C BB                 CMP     E       ;BUT OLD A IS LOST
0A6D C9                 RET
0A6E 414E               DB      'AN'
                ;*
0028 =          SS1:    EQU     28H     ;EXECUTE TIME LOCATION OF THIS INSTRUCTION
0A70 1A                 LDAX    D       ;*** IGNBLK/RST 5 ***
0A71 FE20               CPI     40Q     ;IGNORE BLANKS
0A73 C0                 RNZ             ;IN TEXT (WHERE DE->)
0A74 13                 INX     D       ;AND RETURN THE FIRST
0A75 C32800             JMP     SS1     ;NON-BLANK CHAR. IN A
                ;*
0A78 F1                 POP     PSW     ;*** FINISH/RST 6 ***
0A79 CDCD07             CALL    FIN     ;CHECK END OF COMMAND
0A7C C3E007             JMP     QWHAT   ;PRINT "WHAT?" IFF WRONG
0A7F 47                 DB      'G'
                ;*
0A80 EF                 RST     5        ;*** TSTV OR RST 7 ***
0A81 D640               SUI     100Q    ;TEST VARIABLES
0A83 D8                 RC              ;C:NOT A VARIABLE
0A84 C30301             JMP     TSTV1   ;JUMP AROUND RESERVED AREA
                ; ROUTINE TO COPY RESTART TABLE INTO LOW MEMORY
0008 =          RST1:   EQU     8       ;LOCATION FIRST REATART ROUTINE

0040 =          EOT:    EQU     40H     ;LAST LOC TO BE FILLED

0AA0                    ORG     0AA0H
0AA0 210800     NINIT:  LXI     H,RST1  ;POINT TO BEGINNING OF MODEL TABLE
0AA3 11500A             LXI     D,RSTBL
```

```
0AA6 1A          NXT:    LDAX    D
0AA7 77                  MOV     M,A
0AA8 23                  INX     H
0AA9 13                  INX     D
0AAA 3E40                MVI     A,EOT
0AAC BD                  CMP     L
0AAD C2A60A              JNZ     NXT
0AB0 21FF09              LXI     H,INIT
0AB3 220101              SHLD    START+1
0AB6 C30001              JMP     START
0F00                     ORG     0F00H
0F00 =           TXTEND  EQU     $       ;TEXT SAVE AREA ENDS
0F00             VARBGN  DS      2*27    ;VARIABLE @(0)
0F36                     DS      1       ;EXTRA BYTE FOR BUFFER
0F37             BUFFER  DS      80      ;INPUT BUFFER
0F87 =           BUFEND  EQU     $       ;BUFFER ENDS
0F87                     DS      40      ;EXTRA BYTES FOR STACK
0FAF =           STKLMT  EQU     $       ;TOP LIMIT FOR STACK
2000                     ORG     2000H
2000 =           STACK   EQU     $       ;STACK STARTS HERE
2000                     END
```

```
06F2 ABS        0167 AHOW       080E ASORRY     07E1 AWHAT      0F87 BUFEND
0F37 BUFFER     07A6 CHGSGN     0985 CHKIO      07A3 CHKSGN     0994 CI1
09A8 CI2        07B8 CK1        07B2 CKHLDE     0010 CLOSE      0005 CPM
000E CRLF       09C1 CURRNT     05C2 DEFLT      0013 DELETE     02D5 DIRECT
0786 DIVIDE     0343 DLOAD      097E DONE       038A DSAVE      0791 DV1
0793 DV2        07DC ENDCHK     0040 EOT        07E4 ERROR      02D8 EX0
02DA EX1        02ED EX2        02F7 EX3        02F9 EX4        02FE EX5
02D8 EXEC       05D2 EXPR1      0616 EXPR2      064A EXPR3      06A7 EXPR4
03FC EXRC       005C FCB        03E5 FCBSET     07D4 FI1        07DB FI2
07CD FIN        085F FL1        0874 FL2        03EA FNCLR      0857 FNDLN
085F FNDLNP     0873 FNDNXT     0408 FN         0875 FNDSKP     04A9 FOR
04B9 FR1        04C3 FR2        04C7 FR3        04CA FR4        04CD FR5
04E2 FR7        0503 FR8        0814 GETLN      0818 GL1        083C GL2
0842 GL3        084F GL4        0470 GOSUB      0334 GOTO       016D HOW
0981 IDONE      0559 IFF        09FF INIT       0738 INP        0568 INPERR
09BB INPIO      0572 INPUT      0572 IP1        0580 IP2        0590 IP3
05A9 IP3A       05BA IP4        05C1 IP5        05C8 LET        0420 LIST
035E LOAD       09CB LOPINC     09CD LOPLMT     09CF LOPLN      09D1 LOPPT
09C9 LOPVAR     0429 LS1        0A0E LSTRAM     0980 LSTROM     05D1 LT1
0016 MAKE       0911 MD1        09D8 MSG1       0909 MVDOWN     0900 MVUP
0306 NEW        0508 NEXT       0AA0 NINIT      050F NX0        053F NX1
0551 NX2        0527 NX3        0AA6 NXT        095D OC2        0962 OC3
09C0 OCSW       0172 OK         000F OPEN       09BF OUTCAR     070B OUTCMD
09B0 OUTIO      06BC PARN       0770 PASPRM     0A07 PATLOP     0757 PEEK
08C1 PN1        08C2 PN2        08D2 PN3        08D3 PN4        08DF PN5
08E2 PN6        0747 POKE       0918 POPA       0932 PP1        044C PR0
0454 PR1        0443 PR2        045A PR3        0463 PR6        0467 PR8
0438 PRINT      08ED PRTLN      08B1 PRTNUM     087F PRTSTG     0880 PS1
095A PU1        0934 PUSHA      0166 QHOW       0B0D QSORRY     0890 QT1
0899 QT2        089D QT3        08A5 QT4        08B0 QT5        088B QTSTG
07E0 QWHAT      06E2 RA1        09D3 RANPNT     0381 RDMORE     0014 READD
0555 REM        0490 RETURN     06C7 RND        0008 RST1       0181 RSTART
0A50 RSTBL      0315 RUN        031B RUNNXL     032B RUNSML     0324 RUNTSL
03D9 SAVDON     03AD SAVE       001A SETDMA     07BA SETVAL     06FE SIZE
0703 SIZEA      017B SORRY      0117 SS1A       0028 SS1        0184 ST1
0194 ST2        019D ST3        01A3 ST3A       01D2 ST4        01E3 ST4A
2000 STACK      0100 START      09C3 STKGOS     09C7 STKINP     0FAF STKLMT
030F STOP       079C SUBDE      07CA SV1        01F9 TAB1       021A TAB2
027F TAB4       02B1 TAB5       02B7 TAB6       02BF TAB8       012F TC1
013A TC2        0143 TN1        013E TSTNUM     0103 TSTV1      011F TV1
0124 TV1A       09D7 TXTBGN     0F00 TXTEND     09D5 TXTUNF     0761 USR
0780 USRET      0F00 VARBGN     09C5 VARNXT     071F WAITCM     09B3 WAITIO
0175 WHAT       0015 WRITED     03D1 WRITMOR    05D8 XP11       05DE XP12
05E4 XP13       05EB XP14       05F3 XP15       05F9 XP16       05FF XP17
0601 XP18       061F XP21       0622 XP22       0625 XP23       062C XP24
063D XP25       0640 XP26       064D XP31       0669 XP32       0671 XP33
067C XP34       0699 XP35       06AD XP40       06B6 XP41       06C3 XP42
06C4 XP43
```

Chapter 3

NIBL—National's Tiny Basic Language for SC/MP

By Phil Roybal and Mark Alexander

Since the birth of the first computing machines, users have sought simple ways to build the programs to drive them. While hardware technology advanced by leaps and bounds in the 50's and 60's, man-machine communication struggled painfully forward from patch panels to switch arrays and machine code programming. Finally the first crude assembler, child of a desperate engineer turned programmer, gave promise that users would someday "talk" with their machines in a mutually intelligible language.

Devices available today will allow a computer to identify spoken words and to respond vocally. But it will be a long time before the average hobbyist or industrial user will be able to describe a problem to his computer in common prose and expect any sort of intelligent response. The closest thing to this programmer's dream today is the high-level language: typically a simple, limited vocabulary of written English words in a rigidly defined grammer. The BASIC programming language is a typical example. The high-level language is usually implemented in a translation program (either compiler or interpreter) that analyzes each source code input statement and then expresses it as a series of machine language commands. High-level languages offer these advantages to their users:

- LEGIBILITY—High-level language statements are much easier to read and write than are their assembly language equivalents. This simplifies program construction, debug, and documentation.
- SPEED OF PROGRAMMING—Each statement expands into five to ten machine code commands; thus a high-level language program-

mer can write a program much more quickly than can an assembly language programmer who must generate the same amount of code.

- TRANSPORTABILITY—Programs written in a high-level language are (to some extent) machine independent. Thus code written for one processor can often be used on an entirely different processor with only minor modification.

Of course nothing comes for free, and the advantages of high-level languages have their price. Programs written in them can take 50% to 200% more memory for their operation; and they often execute much more slowly than equivalent assembly language programs. Until quite recently these drawbacks restricted the use of high-level languages to systems where frequent reprogramming was the rule, and some loss of performance was tolerable. They rarely appeared in production computer systems which had to be reproduced in quantity; and they weren't used with microprocessors, which were devoted almost exclusively to the control of such systems. But several things have happened over the last two years to change all that.

First, microprocessors, memories, and related hardware have become simple and inexpensive enough to appeal to the hobbyist, who tends to spend his time programming. He is not interested in production economies since he rarely builds more than one or two systems. But he is highly receptive to tools that make his programming chores easier and more fun.

Second, microprocessor technology has begun to penetrate non-traditional industries. In these new application areas potential users not only have not used programmed logic before, but some have never used electronic components, let alone microcomputers. They have built mechanical and fluid control systems, where the control algorithms were defined by levers and pipes. But they have little electronic experience, and no programming experience at all. They need programming tools that are easy to understand and to use; and they are willing to part with some system speed and economies to get them.

These forces have spurred the development of high-level languages for microprocessors. Thus, when National brought out SC/MP and aimed it at the low-end control market, hobbyists, and non-traditional users, some kind of high-level language seemed like a necessity. And when TINY BASIC a limited version of the standard BASIC language began to gain wide acceptance among hobbyists as a game and control language for microprocessors began to look like a good all around language choice.

For a language to be really useful on microprocessors, it must be good for digital input/output and device control as well as for games and text manipulation. TINY BASIC, developed as a games language, needed improvement in the I/O area. So a series of extensions to the hobbyist standard TINY BASIC were defined and built. Thus was born National Industrial Basic Language: NIBL (pronounced "Nibble") for short. The objectives for the NIBL language were as follows:

- To be a vehicle for rapidly building and evaluating control algorithms.
- To be a simple "self-teaching" approach to the understanding of microprocessors.
- To make use of numerous TINY BASIC games and control programs already in the public domain.
- To be a tool for quickly programming low-volume, moderate-speed application systems.
- To be easily expandable by National and by the user.

TINY BASIC, as originally conceived and written, was an interpreter—a translator that executes source code statements directly, one at a time, without first translating the user's entire program into machine language as compilers do. This approach has the advantage of easier source code manipulation (because the source is always available), and instant feedback of simple grammatical errors. The interpretive approach was retained in NIBL.

Since the original TINY BASIC language was intended to be adapted for many microprocessors, it was itself written interpretively in an ideal language (IL), designed expressly for the construction of translators. Thus, to adapt the language for SC/MP, it was necessary only to construct a simple interpreter that allowed the IL to execute on SC/MP as depicted in the NIBL symbolic concept, Figure 1. The generalized, already written TINY BASIC interpreter would then be usable on SC/MP. This multi-layered, onion-like approach gains two things: the IL interpreter for the NIBL interpreter is smaller and simpler to write than an interpreter for all of NIBL, so the initial investment to produce the product is reduced.

Secondly, since most of NIBL is programmed in a highly efficient instruction set especially tailored for writing translators, the interpreted NIBL is smaller than direct coding would allow. Thus we trade off execution speed for memory efficiency. Though all this sounds complex in the telling, it is invisible to the user, so it doesn't matter to most people.

NIBL LANGUAGE

Language conventions and operating information are shown in the following NIBL Quick Reference Guide. This is a summary of the legal commands and statements of NIBL.

Input Line Format and Editing

- Lines without numbers executed immediately.
- Lines with numbers inserted (in sequence) in program.
- Line number alone (empty line) deletes line.
- Blanks are meaningless, but key words must not contain blanks.
- CONTROL/U deletes current line.
- CONTROL/C aborts program, returns to command mode.
- "←" deletes last character typed.
- CARRIAGE RETURN terminates a line.

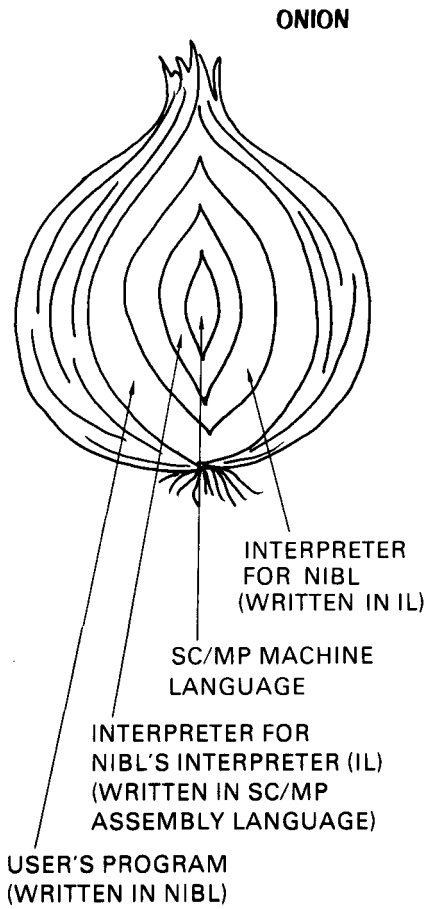


Figure 1. The NIBL Concept. NIBL, an interpreter, is written in an ideal language (IL), which is itself interpreted by a program written in SC/MP assembly language. (Diagram by permission of Dr. Dobb's Journal.)

Commands

CLEAR deletes all program information
RUN runs program
LIST lists program
LIST X lists program from line X on
GOTO X runs program from line X on
PAGE = X selects 4K byte user page X
NEW clears program but not variable assignments from the selected page.

Statements

Input/Output Statements

```
PRINT X,Y,Z
PRINT "THE ANSWER IS," X
INPUT X,Y,Z
aL = 1
J = aL
```

Assignment Statements

```
LET X = 3
LET X = -15 + 5*Y
```

Control Statements

```
GOTO X + 10
GOTO 10
GOSUB Y + 15
RETURN
IF X = Y THEN GOTO 15
DO: A = A + 1: PRINT A: UNTIL A = 10
FOR B = 2 TO 20 STEP 2
....
....  NEXT B
....
REM THIS IS A COMMENT LINE
END Ends a program
LINK 5000
```

Variables

A.....Z (26 only)

Constants

Numbers

Decimal 0.....9
HEX 0.....F
All arithmetic is module 2^{15} (± 32768)

Operators

Arithmetic	Relational	Logical
+ -	>>=	OR
*/	<<=	AND
	=<>	NOT

NIBL GRAMMAR

As originally defined, TINY BASIC had the grammatical elements indicated by the asterisk sign (*) in the text margin. While this grammar is adequate for a variety of games and internal data manipulations, it

can be improved considerably for control applications through the addition of the rest of the elements shown in the following:

```
NIBL LINE = IMMEDIATE STATEMENT
          ! PROGRAM LINE
          ! INPUT RESPONSE ;
IMMEDIATE STATEMENT =
(COMMAND ! STATEMENT) CARRIAGE RETURN ;
PROGRAM LINE =(DECIMAL NUMBER STATEMENT LIST
          CARRIAGE RETURN) ;
INPUT RESPONSE =REL EXP (',' REL EXP) * CARRIAGE
          RETURN ;
  COMMAND = 'NEW'
*   ! 'CLEAR'
*   ! 'LIST' DECIMAL NUMBER?
*   ! 'RUN'
*   ! 'PAGE'
    ;
  STATEMENT LIST = STATEMENT
    ';' STATEMENT) * ;
* STATEMENT = 'LET' ? (VARIABLE ! 'STAT' ! 'a' FACTOR)
    '=' REL EXP
*   ! 'GO' ('TO' ! 'SUB') REL EXP
*   ! 'RETURN'
*   ! ('PR' ! 'PRINT') PRINTLIST
*   ! 'IF' REL EXPR 'THEN' ? STATEMENT
*   ! 'DO'
    ! 'UNTIL' REL EXP
    ! 'FOR' VARIABLE '=' REL EXP 'TO' REL EXP
                                ('STEP' REL EXP) ?
    ! 'NEXT' VARIABLE
*   ! 'INPUT' VARIABLE +
    ! 'LINK' REL EXP
    ! 'REM' ANY CHARACTER EXCEPT CARRIAGE
      RETURN
*   ! 'END'
    ;
  REL EXPR = EXPRESSION RELOP EXPRESSION
    ! EXPRESSION
    ;
*   RELOP = '<'
*   ! '<' '='
*   ! '<' '>'
*   ! '>'
*   ! '>' '='
*   ! '='
    ;
    EXPRESSION = EXPRESSION ADDING
      OPERATOR TERM
```

```
      ;
        ADDING OPERATOR = '+' ! '-'
          ! 'OR' ;
  TERM = TERM MULTIPLYING OPERATOR FACTOR
          ! FACTOR
          ;
* MULTIPLYING OPERATOR = '*' ! '/'
          ! 'AND' ;
      *   FACTOR = VARIABLE
      *   ! DECIMAL NUMBER
      *   ! '<' REL EXPR '>'
          ! 'a' FACTOR
          ! '#' HEX NUMBER
          ! 'NOT' FACTOR
          ! 'RND' '<' REL EXP ',' REL EXP '>'
          ! 'STAT'
          ! 'TOP'
          ;
      *   VARIABLE = 'A' ! 'B' ! 'C' ! . . . ! 'Y' ! 'Z' ;
      *   DECIMAL NUMBER = DECIMAL DIGIT + ;
      *   DECIMAL DIGIT = '0' ! '1' ! '2' ! . . . ! '9' ;
          HEX NUMBER = (DECIMAL DIGIT ! HEX DIGIT) +
          HEX DIGIT = 'A' ! 'B' ! 'C' ! 'D' ! 'E' ! 'F' ;
          PRINTLIST = (STRING ! REL EXPR) (STRING ! REL
            EXPR) * ';' ? ;
          STRING = '"' ALMOST ANY CHARACTER '"'
```

NOTE: ALL ITEMS IN SINGLE QUOTES ARE ACTUAL SYMBOLS IN NIBL; ALL OTHER IDENTIFIERS ARE SYMBOLS IN THE GRAMMAR. THE EQUALS SIGN (=) MEANS "IS DEFINED AS"; PARENTHESES ARE USED TO GROUP SEVERAL ITEMS TOGETHER AS ONE ITEM; THE EXCLAMATION POINT (!) MEANS AN EXCLUSIVE-OR-CHOICE BETWEEN THE ITEMS ON EITHER SIDE OF IT; THE ASTERISK (*) MEANS ZERO OR MORE OCCURRENCES OF THE ITEM TO ITS LEFT; THE PLUS SIGN (+) MEANS ONE OR MORE REPETITIONS; THE QUESTION MARK (?) MEANS ZERO OR ONE OCCURRENCES; AND THE SEMICOLON (;) MARKS THE END OF A DEFINITION.

IN NIBL, SPACES ARE NOT SIGNIFICANT EXCEPT WITHIN THE SYMBOLS IN SINGLE QUOTES IN THE GRAMMAR ABOVE. NUMBERS AND RESERVED WORDS LIKE 'PRINT' OR 'GO' MUST NOT HAVE SPACES WITHIN THEM. A VARIABLE MUST BE FOLLOWED IMMEDIATELY BY A NON-ALPHABETIC CHARACTER IN ORDER TO DISTINGUISH IT FROM A KEY WORD.

System Requirements

The NIBL interpreter is intended to be a ROM-resident program in the first 4K of the SC/MP Memory address space (although it will run just as well in RAM). The interpreter requires at least 2K bytes of RAM starting at address 1000 (base 16), of which the interpreter uses nearly 300 bytes for stacks, variables, etc., leaving the rest for the user's program. Another 2K bytes of memory may be added to fill up this 4K page, forming what is hereafter referred to as "Page 1."

The SC/MP architecture forces memory to be split into pages of 4K bytes each; therefore, NIBL allows seven such pages to be used for storing programs. NIBL programs in the seven pages are edited separately, but may be linked together during program execution by special NIBL statements described in the following pages. Page #1 as mentioned above, must be RAM since the interpreter uses part of it as temporary storage; the part used to store application programs starts at location 111E (base 16).

The other six pages of memory, each of which starts at location n000 (base 16), where n is the page number, may be either RAM or ROM. Page 2 is a special page: it can contain a NIBL program to be executed immediately upon powering up the NIBL system.

The memory organization of NIBL is shown in Figure 2.

Throughout this article, the assumption is made that the user has a teletype with paper tape reader and punch, as with the SC/MP Low Cost Development System (LCDS). In fact, NIBL was designed to use the SC/MP LCDS teletype interface, but to be completely independent of the LCDS firmware. If NIBL is to be run on its own, the system should have the same configuration for the teletype, with the reader relay being operated directly by the SC/MP. At present, paper tape is the only medium for saving NIBL programs, but as soon as the hardware and software for SC/MP cassette interface become available, NIBL will be able to link to routines for saving and loading programs with ease.

Since the teletype interface is not based on a UART, the terminal baud rate can only be changed by modifying the time delays in NIBL's I/O routines. NIBL has been run successfully at 1200 baud with a CRT terminal; the assembly listing of the NIBL program is programmed for a 110 baud rate system.

Communicating with NIBL

When the NIBL system is ready to accept input, it prompts at the teletype with a ">" prompt sign. (NIBL is now in "edit mode.") The user then enters a line terminated by a carriage return. There are several special characters that are used to edit lines as they are typed:

Shift/O (back-arrow) causes the last character type to be deleted.

Control/H (backspace) performs the same function as shift/O, but echoes as a backspace/space/backspace sequence, which is only useful if the teletype routines are modified to run at high baud rates for use with CRT's.

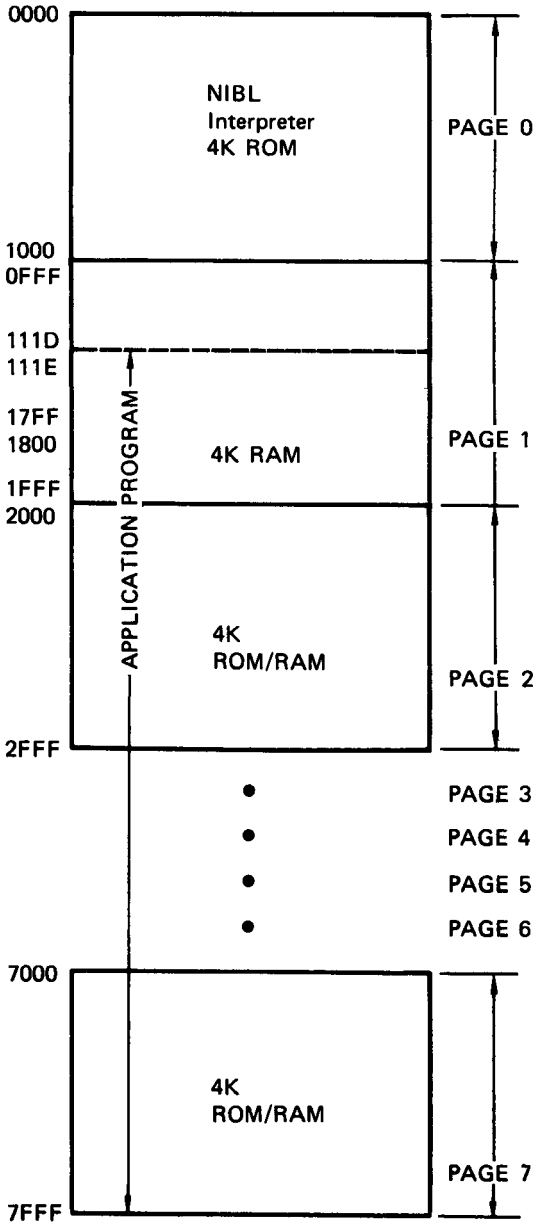


Figure 2. NIBL Memory Organization

Control/U (echoes as "'U") causes the entire line to be deleted; NIBL reprompts for a new line.

Entering a line to NIBL without a leading line number causes the line to be executed directly by NIBL. Most NIBL statements, as well as the four program control commands, may be executed in this manner.

A line with a leading number (in the range 0 through 32767) is entered into the NIBL program in the current page. (Make sure that the value of the pseudo-variable PAGE is valid, so that the line isn't into non-existent memory.) The NIBL editor sorts the program lines as they are entered into ascending order by line number.

Typing a line number followed by a carriage return deletes that line from the program. Typing a line with the same number as an existing line's causes the new line to replace the old one in the program.

Each of the seven memory pages may contain a different program, separate from the rest. Editing the program in one page will not affect the other pages. To switch editing from one page to another, simply type PAGE = n, where n is the number of the new page.

Variables

There are twenty-six variable names in NIBL: the letters A through Z. They are all 16-bit binary variables, so they can be used to hold addresses as well as signed numeric data. Space for pre-declared variables is allocated for them in RAM when NIBL powers up.

Constants

NIBL allows either decimal or hexadecimal (base 16) constants to appear in expressions. Decimal constants must lie in the range 0 through 32767; the unary minus ("–") is used to obtain negative values. The value –32768 is a valid NIBL integer, but it is not legal as it stands. To represent it, use –32767-1 or #8000 instead.

Hexadecimal constants are denoted by a pound sign ("#") followed by a string of hexadecimal digits (0-9, A-F). NIBL does not check for overrun in hex constants; consequently, only the 4 least significant digits of the hex digit string are kept.

Functions

NIBL includes three functions that may appear in any expression. These are described as follows:

RND (X, Y) returns a pseudo-random integer in the range X through Y, inclusive, where X and Y are arbitrary expressions. In order for the function to work properly, the value of Y – X should be positive and no greater than 32767.

MOD (X, Y) returns the absolute value of the remainder from X divided by Y (where X and Y are expressions).

TOP (with no arguments) returns the address of the first free byte in the memory page currently being edited or executed. In other words, it is the address of the last byte at the top of the current NIBL application program in the current page, plus one.

Pseudo-Variables

NIBL has two pseudo-variables in addition to the standard variables. These are STAT and PAGE. Both of these variables may appear on either side of an assignment statement.

STAT represents the SC/MP status register. The current value of the status register can be referred to by using STAT in an expression; or an assignment may be made to the status register by executing a statement such as STAT = 4 or STAT = STAT OR #20. When NIBL makes an assignment to the status register in this manner, it clears the interrupt-enable bit of the value before it is actually assigned. Note also that only the lower byte of the value is assigned, the high byte is ignored.

The carry and overflow bits in STAT are meaningless since the NIBL system is continually modifying them. The utility of STAT lies in the fact that 5 of its bits are connected to I/O sense lines on the SC/MP chip.

The pseudo-variable PAGE contains the number of the memory page currently being executed or edited. As indicated in Figure 2, there are seven pages in which NIBL programs may be stored; therefore, the PAGE number may be only in the range of 1 through 7. If an assignment of a value outside this range is made to PAGE, only the 3 least significant bits LSBs of the value are used. Page one is defined as LSBs = 000, page two as LSBs = 001 etc.

If PAGE is modified while NIBL is in edit mode, all subsequent editing will take place in the new page.

If PAGE is modified by a NIBL program during execution, control will be passed to the first line of the NIBL program in the new page. This transfer would be effected by a statement such as PAGE = 6 or PAGE = PAGE + 1. Thus, several NIBL programs residing in different 4K pages may be linked together as one large program, if need be. This would allow one to write a 28K STAR TREK program in NIBL, a Herculean and indeed foolish task.

Control may also be transferred from one page to another by three other statements: RETURN, NEXT, and UNTIL. Thus, the first part of a subroutine or loop may be in one page, and the second part may be in another (with control being transferred between the two parts by an assignment to PAGE). In these three special cases, NIBL automatically updates the value of PAGE as the statements are executed.

Relational Operators

NIBL provides the standard BASIC relational operators, for comparing the values of integer expressions. The operators are as follows:

```
  =  equal to
 <=  less than or equal to
 >=  greater than or equal to
 <>  not equal to
  <  less than
  >  greater than
```

All of these operators produce 1 as a result if the relation is true, and 0 if the relation is false. Note that the relational operators may appear anywhere that an expression is called for in the NIBL grammar, not only in IF statements.

Arithmetic Operators

NIBL provides the four standard arithmetic functions: Addition (+), subtraction or unary minus (−), multiplication (*/), and division (/). Since only integers are allowed in NIBL, all quotients are truncated (the MOD function can be used to obtain remainders from division). Any overflow or underflow (other than division by zero) is ignored by NIBL; the reasoning behind this is that it may often be necessary to treat NIBL expressions as unsigned values, such as when performing calculations using memory addresses as the operands. Thus the value of 32767 + 1 is −32768 (or in hexadecimal, °7FFF + 1 = #8000, which makes more sense).

Logical Operators

In NIBL, there are three logical operations that may be performed on values; AND, OR, and NOT. The first two are binary operators, and the latter is unary. All three perform bitwise logical operations on 16-bit arguments, producing 16-bit results. AND, OR, and NOT are sufficient to simulate any other logical operation, through various combinations of the operators.

The Indirect Operator

The indirect operator "@" realizes the functions of PEEK and POKE operations in other BASICs, but with somewhat more elegance. The "@" sign followed by an address (which can be a constant, variable, or expression in parentheses) denotes the contents of that address in memory. Thus, if memory location 245 (decimal) contains 60, the statement X = @245 would result in the value 60 being assigned to X. The indirect operator may also appear on the left side of an assignment statement. For example, @X = @(Y + 10) would result in the memory location pointed to by X being assigned the value of the memory location pointed by the value Y + 10.

Use of the indirect operator is not limited to reading from or writing to memory: it also provides a simple way to communicate with peripheral devices that are interfaced to the SC/MP through memory addresses. Note that the "@" operator can only access memory one byte at a time, and that when an assignment is made to a memory location, only the low order byte of the value is moved to the location; the high order byte is ignored.

The indirect operator can also be used to simulate arrays in NIBL. For example, if we wish to define an M × N matrix of one-byte positive integers, we can access the (I,J)th element of the matrix (assuming that (0,0) is a legal element in the matrix) with the expression @(A + I*N + J). An assignment could be made to that same element by placing the expression on the left side of an assignment statement.

Expressions

Expressions in NIBL are composed of variables, constants, function references, pseudo-variables, and operators. Operators bind these other elements together to form complete expressions. NIBL expressions are all 16-bit integers. Evaluation of expressions takes place left-to-right, and the order in which operations take place is determined by operator precedence and the presence of parentheses. The order of evaluation can be deduced from the grammar in Table 1; here is a table of operator precedence:

Lowest precedence (applied last): <,>,<=,>=,=,<>
+,-,OR
*,/,AND

Highest precedence (applied first): @, NOT

Despite this, it is still safest to use plenty of parentheses in expressions to make the intent clear.

Program Control Commands

LIST causes the entire program in the current page to be listed. Listing can be halted by hitting any key on the teletype: the BREAK Key works best.

LIST <number> causes listing to begin at the given line number (or the nearest one greater than the number), rather than at the first line.

LISTing a program is the method used to save it on paper tape. To accomplish this, type LIST with the punch off, then turn on the punch and hit carriage return. After the program is dumped, type a Shift/O with teletype on LOCAL so that the last character (a ">") will be deleted when the tape is entered to NIBL at a later time. NIBL will accept a tape made in this fashion at any time during edit mode. The tape reader is enabled at all times by NIBL, and it does not distinguish between the reader and the keyboard when accepting input. Superfluous line-feed and null characters on the tape are echoed but ignored.

RUN causes three actions: first, all variables are zeroed; secondly, all stacks (the FOR, DO, and GOSUB stacks) are cleared; and finally the program in the current page is executed, starting with the first line in sequence.

RUN is not the only way to start program execution: GOTO and GOSUB can also be used to jump into a program from edit mode. For example, if there is a subroutine at line 1000 that is being tested, typing GOSUB 1000 will cause that routine to be executed, with NIBL returning to edit mode upon encountering a RETURN statement. When GOTO and GOSUB are used to run a program, the variables and stacks are not cleared.

Hitting any key while a program is being run will cause NIBL to break execution, printing a message and the line number where the break was detected. The BREAK key on the teletype works best for this.

CLEAR causes all variables to be zeroed and the three stacks mentioned above to be cleared. This latter feature of the CLEAR command is quite useful after a stack nesting error has occurred (for example, if GOSUBs are nested more than eight levels deep).

NEW clears the program in Page 1, and changes the value of PAGE to 1. This is the form of the command most likely to be used by NIBL novices who do not wish to be confused by the page selection features of NIBL. NEW should be the first thing one types in to NIBL when first powering up.

NEW <number> sets the value of PAGE to the <number>, and clears the program in that page.

Assignment Statements

Previously, two different types of assignment statements have been described: assignments to the pseudo-variables STAT and PAGE, and assignments to memory locations with the indirect operator. Another form of the assignment statement is the conventional assignment to a variable (A = Z), e.g. A = A + 1 or A = 32<(4*I). There are also statements which look like string assignments, but these are not standard BASIC, and are described later in the section on string handling. The word "LET" is optional in front of any assignment statement (leaving it out increases execution speed, unlike most Tiny BASIC systems).

IF/THEN Statement

The IF statement allows conditional execution of one or more statements (as many as can fit on one line). The syntax for the IF statement is:

'IF' Rel-exp 'THEN'? Statement

which indicates that the word THEN is optional, and that any statement (including another IF statement) may follow the conditional expression. If the IF condition is true (i.e. is non-zero), the statement following it (and any others on the line) will be executed; otherwise, control immediately transfers to the next program line. The condition does not need to contain relational operators: a statement such as IF MOD (A,5) THEN. . . is perfectly legal. In this example, the statement following the THEN would be executed if A were not divisible by 5.

GOTO, GOSUB, and RETURN Statements

The syntax for the GOTO statement is 'GOTO' followed by an expression. The effect of the GOTO statement is to transfer control to the line whose number is indicated by the expression. An error occurs if the specific line does not exist in the current page. Unlike standard BASICs, any arbitrary expression can be used to specify the line number, as well as the usual decimal constant. This allows computed branches to be eprformed with the same effect as the ON. . . GOTO statement in standard BASIC.

The GOSUB statement is identical to the GOTO statement in form. It too causes a branch to a new line, but it also saves the address of the following statement on a stack. When a RETURN statement is executed, the saved address is popped from the stack, and control returns to that point in the program. Since an actual address, not a line number, is saved on the GOSUB stack, GOSUB statements may appear anywhere on a multiple-statement line.

GOSUBs may be nested up to eight levels deep; an error will occur if an attempt is made to exceed this limit. The error condition does not destroy the previous contents of the stack, so a RETURN statement can be executed (even in edit mode) without an error occurring. However, any modification of the NIBL program will clear the GOSUB stack, so that a subsequent RETURN without a GOSUB will cause an error.

DO and UNTIL Statements

The DO and UNTIL statements are useful in writing program loops efficiently, without using misleading GOTO statements. Enclosing a group of zero or more statements between a DO statement and an UNTIL <condition> statement (where <condition> is an arbitrary expression) will cause the statement group to be repeated one or more times until the <condition> becomes true (i.e. non-zero). As an example of the use of the DO and UNTIL statements, we present a program that prints the prime numbers:

```
 10 PRINT 1: PRINT 2
 20 I = 3
 30 DO
 40 J = I/2: N = 2
 50 DO
 60 N = N + 2
 70 UNTIL (MOD (I,N) = 0) OR (N>J)
 80 IF N>J PRINT I
 90 I = I + 2
100 UNTIL 0
```

DO loops may be nested up to eight levels deep, and NIBL acts in the same manner if an overflow occurs as it does with a GOSUB overflow. NIBL also reports an error if an UNTIL statement occurs without a previous DO. A single DO loop may have more than one UNTIL statement as a terminator. For example, if one wished to exit abnormally out of a DO loop and transfer to some appropriate line, it could be done in the following manner:

```
UNTIL 1: GOTO X
```

where X is the line number.

Neither the DO nor the UNTIL statement may be executed in edit mode.

FOR and NEXT Statements

The NIBL FOR statement is virtually identical to that in standard BASICs; consequently, it is not explained in great detail here.

As in most BASICs, both positive and negative STEPs are allowed in the FOR statement, and a STEP of +1 is assumed if the STEP portion of the statement is omitted. A FOR Loop is terminated by a NEXT <variable> statement, and the <variable> must be the same as that referred to in the FOR statement at the beginning of the loop.

FOR loops may be nested four levels deep; NIBL reports an error if this limit is exceeded, or if a NEXT statement occurs without a previous FOR statement. As with the DO and UNTIL statements, FOR and NEXT may not be executed in edit mode.

Perhaps the only differences between the NIBL FOR statement and that of more elaborate BASICs (such as DEC's BASIC-PLUS for the PDP-11) are that a FOR Loop is always executed at least once, and that when a NEXT statement is executed, the STEP value is added to the variable before the test is made to determine if the loop should be repeated (rather than after the test).

INPUT Statement

There are two types of INPUT statements in NIBL: numeric input and string input. The form of the first type is 'INPUT' followed by a list of one or more variables. When this statement is executed, NIBL prompts at the teletype with a question mark ("?"). The user responds with a list of expressions separated by commas, and terminated by a carriage return. For example, a legal response to the statement INPUT A,B,C would be #3FA,26,4*27. These three expressions would then be assigned to the variables A,B, and C, respectively. An illegal response (too few arguments or improper expressions) will result in a syntax error. Any extra arguments in the response are ignored.

The second type of INPUT statement allows strings to be input. The form of the statement is "INPUT' '$' <address>, where <address> is a Factor, syntactically (usually a variable, constant, or expression in parentheses). When this statement is executed, NIBL prompts the user as before, at which point the user enters a line terminated by the usual carriage return. NIBL then stores the line in memory in consecutive locations, beginning at the address specified. Thus, INPUTS #6000 would cause the input line to be stored starting at location 6000 (base 16); the carriage return would also be stored at the end of the line.

Strings input in this manner can be tested and manipulated by using the "@" operator or the string handling statements described below. They can also be displayed by a PRINT statement.

Neither of the two INPUT statements may be executed in edit mode.

PRINT Statement

The form of the PRINT statement is 'PRINT' or 'PR' followed by a list of print items separated by commas, and optionally terminated by a

semicolon, which suppresses an otherwise automatic carriage return after all items in the list are printed.

A print item consists of one of the following:

- A quoted string, which is printed exactly as it appears (with the quotes removed)
- An expression, which is evaluated and printed in decimal format, with either a leading space or a minus sign (" – "), and one trailing space
- A reference to a string in memory, denoted by '$' <address>, where <address> is a Factor as usual. Successive memory locations, starting at the specified address, are printed as ASCII characters, until a carriage return (which is not printed) is encountered.

There is no zone spacing in the PRINT statement, nor does NIBL perform an automatic carriage return/line feed after printing 72 characters. NIBL is not an output-oriented language; fancy formatting has been sacrificed for more useful control structures and data manipulation features. (A subroutine to print a number and skip to the next print zone is a trivial to write in NIBL—it takes about two lines of code, with the DO/UNTIL and FOR/NEXT.)

String Handling Statements

String handling in NIBL is very minimal and low-level. The string handling features of the INPUT and PRINT statements have already been mentioned; NIBL provides two more statements for manipulating strings.

A statement such as $<address> = "THIS IS A STRING" would cause the quoted string to be stored in memory starting at the specified address (which again is a Factor), with a carriage return being appended to the string.

Another statement allows the programmer to move strings around in memory once they have been created. The form of this statement is '$' <destination> '=' '$' <source>, where both <destination> and <source> are Factors, and are the addresses of strings in memory. This statement causes all the characters in the string pointed to by <source> to be copied one-by-one to the memory pointed to by <destination>, until a carriage return (also copied) is encountered. Overlapping the source and destination addresses can produce disastrous results, such as wiping out the entire contents of the current page. Consequently, a string move can be aborted by hitting the BREAK key on the teletype (but it must be done quickly!).

Note that all strings referred to in these statements, and in the INPUT and PRINT statements, are assumed to lie within a 4K page, and wraparound is a possibility which must be anticipated by the programmer. (Long-time SC/MP programmers will be familiar with this minor problem.)

Using these statements, it should be very easy to develop a set of NIBL subroutines for performing concatenation, comparison, and substring operations on strings.

END Statement

The END statement may appear anywhere in a NIBL program and not necessarily at the end. It causes a message and the current line number to be printed, with NIBL returning to edit mode. The END statement is useful when debugging programs, since it acts as a breakpoint in the program that can be removed easily.

LINK Statement

The LINK statement allows NIBL programs to call SC/MP machine language routines at any address. A statement of the form 'LINK' <address>, where <address> is an arbitrary expression, will cause the NIBL system to call the routine at that address by executing an appropriate XPPC P3 (Exchange contents of the Program Counter with designated Pointer Register P3) instruction. The user's routine should make sure that it returns by executing another XPPC P3, and that the value of P3 (Pointer Register #3) upon entry to the routine is restored before returning. The routine may make use of the fact that P2 is set by NIBL to point to the beginning of the RAM block used to store the variables A through Z, with each variable being stored low byte first, high byte second. Thus, parameters may be passed between NIBL programs and machine language routines through the variables. Both P1 and P2 may be modified by the user's routines; they are automatically restored by the NIBL system upon return. The user should be careful not to modify RAM locations with negative displacements relative to P2, or the locations with displacements greater than 51 relative to P2. These locations are used by the interpreter.

Remark Statement

A comment can be inserted into a NIBL program by preceding it with the word REM, REM causes the rest of the line to be ignored by NIBL during execution. Remarks are useful in debugging programs or helping other people to understand them, but of course, they take up valuable memory. (Then again, memory is getting cheaper all the time.)

Multiple Statements On One Line

A program line may contain more than one statement, if the statements are separated by colons (";"). Using multiple statements on a single line improves the readability of the program by separating it into small blocks, and uses less memory for storing the program.

It is important to note that an IF statement will cause any statements appearing after it on the line to be ignored if the IF condition turns out to be false. This is the feature that allows a group of statements to be executed conditionally.

A multiple-statement line may be entered without a line number, but NIBL will only execute the first statement on the line, ignoring the rest.

NIBL Applications

Most users sitting at a NIBL terminal for the first time will want to start playing games with the machine. There seems to be nothing like the thrill of having some quasilogical conversation with, and being soundly beaten by, a computer. TINY BASIC was originally built to implement games; and it performs well in this mode. An example of a simple NIBL game which will illustrate the flavor of the language is shown in Listing 1.

Listing 1. "Bagels"—A Simple NIBL Game

```
10 PRINT "      BAGELS":
   PRINT "":
   PRINT ""
40 PRINT " I WILL THINK OF A THREE DIGIT NUMBER. YOU TRY TO"
50 PRINT " GUESS WHAT IT IS.  FOR EACH CORRECT DIGIT IN THE"
53 PRINT " CORRECT LOCATION, I WILL PRINT 'FERMI'.  FOR EACH"
55 PRINT " CORRECT DIGIT NOT IN THE CORRECT LOCATION, I WILL"
57 PRINT " PRINT 'PICO'.  IF NO DIGITS ARE CORRECT, I WILL PRINT"
58 PRINT " 'BAGLES'.":
   PRINT "":
   PRINT ""
59 REM
60 A=RND(1,9):
   B=RND(0,9):
   C=RND(0,9):
   P=0
70 REM SELECT A NUMBER
71 REM
120 PRINT " PLEASE GUESS A THREE DIGIT NUMBER.";
130 INPUT G:
   REM INPUT GUESS, TEST RANGE
135 REM
140 IF G> 1000 OR G<100 GOTO 120
160 M=0:
   N=0:
   P=P+1:
   H=G/100:
   REM ZERO CNTRS, SELECT LEFT DIGIT
200 IF H=A M=M+1:
     REM CORRECT DIGIT & LOCATION
210 IF ((H=B) OR (H=C)) N=N+1:
     REM CORRECT DIGIT, BAD LOCATION
230 I=MOD(G,100)/10:
   REM SELECT MID. DIGIT OF INPUT
```

```
240 IF ((I=A) OR (I=C)) N=N+1:
      REM CORRECT DIGIT, BAD LOCATION
250 IF I=B M=M+1:
      REM CORRECT DIGIT & LOCATION
270 J=MOD(G,10):
    REM SELECT RIGHT DIGIT OF INPUT
280 IF ((J=A) OR (J=B)) N=N+1:
      REM CORRECT DIGIT, BAD LOCATION
300 IF J=C M=M+1:
      REM CORRECT DIGIT & LOCATION
310 IF M<3 GOTO 600
320 PRINT " CONGRATULATIONS! YOU GOT IT IN", P ,"TRIES."
330 PRINT " PLAY AGAIN ?  (1=YES, 0=NO )"
340 INPUT Q:
    IF Q=0 GOTO 1000
360 GOTO 60
500 REM
550 REM NEXT SECTION PRINTS CLUES
600 IF M<>0
      FOR T=1 TO M:
        PRINT "FERMI  ";:
      NEXT T
620 IF N<>0
      FOR T=1 TO N:
        PRINT "PICO  ";:
      NEXT T
650 IF M+N =0 PRINT "BAGLES"
700 PRINT "":
    GOTO 120:
    REM ASK FOR NEXT GUESS
1000 PRINT "GOODBYE"
```

The object of the game is to guess the number that the microprocessor has picked. All numbers are between 100 and 999. For each correctly guessed digit in the correct location, the processor responds "FERMI." For each correct digit not in the right location, the processor responds "PICO." If no correct digits are guessed, the processor responds "BAGLES."

The NIBL Language is well suited to control tasks, as long as the user recognizes its inherent speed limitations. While it is more than adequate for human interface and a variety of other control applications, it doesn't have the speed to handle video generation, direct control of fast peripherals, etc. For these applications, the algorthims should be proved out in NIBL, then translated into SC/MP machine code for installation in the final system. On the plus side, once the user has paid the initial price in speed and ROM for the interpreter, he will find that NIBL tasks (which are stored as powerful source

statements) tend to take less memory than their assembly language equivalents. The larger the program, the more dramatic are the savings.

Bibliography

DR. DOBB'S JOURNAL OF COMPUTER CALISTHENICS AND ORTHODONTIA; Volume 1, No. 1—January, 1976. PCC, Box 310, Menlo Park, CA 94025.

And thanks to Dr. Marvin Winzinread, California State University at Hayward, for "BAGLES."

```
1
2       0001              LIST   1
3
4               ;*************************************************
5               ;*                                               *
6               ;*      WE ARE TIED DOWN TO A LANGUAGE WHICH     *
7               ;*      MAKES UP IN OBSCURITY WHAT IT LACKS      *
8               ;*      IN STYLE.                                *
9               ;*                      -- TOM STOPPARD          *
10              ;*                                               *
11              ;*************************************************
12
13      0020    TSTBIT  =       020             ;I.L. INSTRUCTION FLAGS
14      0040    JMPBIT  =       040
15      0080    CALBIT  =       080
16      0001    P1      =       1               ;SC/MP POINTER ASSIGNMENTS
17      0002    P2      =       2
18      0003    P3      =       3
19      FF80    EREG    =       -128            ;THE EXTENSION REGISTER
20
21              ; DISPLACEMENTS FOR RAM VARIABLES USED BY INTERPRETER
22
23      FFFF    DOPTR   =       -1              ;DO-STACK POINTER
24      FFFE    FORPTR  =       -2              ;FOR-STACK POINTER
25      FFFD    LSTK    =       -3              ;ARITHMETIC STACK POINTER
26      FFFC    SBRPTR  =       -4              ;GOSUB STACK POINTER
27      FFFB    PCLOW   =       -5              ;I.L. PROGRAM COUNTER
28      FFFA    PCHIGH  =       -6
29      FFF9    PCSTK   =       -7              ;I.L. CALL STACK POINTER
30      FFF8    LOLINE  =       -8              ;CURRENT LINE NUMBER
31      FFF7    HILINE  =       -9
32      FFF6    PAGE    =       -10             ;VALUE OF CURRENT PAGE
33      FFF5    LISTNG  =       -11             ;LISTING FLAG
34      FFF4    RUNMOD  =       -12             ;RUN/EDIT FLAG
35      FFF3    LABLLO  =       -13
36      FFF2    LABLHI  =       -14
37      FFF1    P1LOW   =       -15             ;SPACE TO SAVE CURSOR
38      FFF0    P1HIGH  =       -16
39      FFEF    LO      =       -17
40      FFEE    HI      =       -18
41      FFED    FAILLO  =       -19
42      FFEC    FAILHI  =       -20
43      FFEB    NUM     =       -21
44      FFEA    TEMP    =       -22
45      FFE9    TEMP2   =       -23
46      FFE8    TEMP3   =       -24
47      FFE7    CHRNUM  =       -25
48      FFE6    RNDF    =       -26
49      FFE5    RNDX    =       -27             ;SEEDS FOR RANDOM NUMBER
50      FFE4    RNDY    =       -28
51
52              ; ALLOCATION OF RAM FOR NIBL VARIABLES, STACKS,
53              ;  AND LINE BUFFER
54
```

```
                        .=01000+28
101C        VARS:       .=.+52                  ; NIBL VARIABLES A-Z
1050        AESTK:      .=.+26                  ; ARITHMETIC STACK
106A        SBRSTK:     .=.+16                  ;GOSUB STACK
107A        DOSTAK:     .=.+16                  ;DO/UNTIL STACK
108A        FORSTK:     .=.+28                  ;FOR/NEXT STACK
10A6        PCSTAK:     .=.+48                  ;I.L. CALL STACK
10D6        LBUF:       .=.+74                  ;LINE BUFFER
1120        PGM:        .=0                     ;USER'S PROGRAM

                        .MACRO  LDPI,P,VAL
                        .MLOC   TEMP
                        .SET    TEMP,VAL
                        LDI     H(TEMP)
                        XPAH    P
                        LDI     L(TEMP)
                        XPAL    P
                         ENDM


                    ;************************************
                    ;*       INITIALIZATION OF NIBL     *
                    ;************************************

0000 08                 NOP
0001                    LDPI    P2,VARS         ;POINT P2 AT VARIABLES
0007                    LDPI    P1,PGM          ;POINT P1 AT PAGE ONE PROGRAM
000D C4FF               LDI     -1              ;STORE -1 AT START OF PROGRAM
000F C900               ST      0(P1)
0011 C901               ST      1(P1)
0013 C40D               LDI     0D              ;ALSO STORE A DUMMY END-OF-LI
0015 C9FF               ST      -1(P1)
0017 C402               LDI     2               ;POINT P2 AT PAGE 2,
0019 CAF6               ST      PAGE(P2)        ; INITIALLY SET PAGE TO 2
001B 31                 XPAL    P1
001C C420               LDI     020
001E 35                 XPAH    P1
001F B902               DLD     2(P1)           ;CHECK IF THERE IS REALLY
0021 01                 XAE                     ; A PROGRAM IN PAGE 2:
0022 C180               LD      EREG(P1)        ; IF FIRST LINE LENGTH
0024 E40D               XRI     0D              ; POINTS TO CARR. RETURN
0026 9802               JZ      $0              ; AT END OF LINE
0028 BAF6               DLD     PAGE(P2)        ;IF NOT, PAGE = 1
002A C420   $0:         LDI     020
002C 35     $LOOP:      XPAH    P1
002D C4FF               LDI     -1              ;STORE -1 IN 2 CONSECUTIVE
002F C900               ST      (P1)            ; LOCATIONS AT START OF PAGE
0031 C901               ST      1(P1)
0033 C40D               LDI     0D              ;ALSO PUT A DUMMY END-OF-LINE
0035 C9FF               ST      -1(P1)          ; JUST BEFORE TEXT
0037 35                 XPAH    P1              ;UPDATE P1 TO POINT TO
0038 02                 CCL                     ; NEXT PAGE (UNTIL PAGE=8)
0039 F410               ADI     010             ;REPEAT INITIALIZATION
003B E480               XRI     080             ; FOR PAGES 2-7
003D 9804               JZ      $1
003F E480               XRI     080
0041 90E9               JMP     $LOOP
0043 C400   $1:         LDI     0               ;CLEAR SOME FLAGS
0045 CAF4               ST      RUNMOD(P2)
0047 CAF5               ST      LISTNG(P2)
0049 C454               LDI     L(BEGIN)        ;INITIALIZE IL PC SO THAT
004B CAFB               ST      PCLOW(P2)       ; NIBL PROGRAM
004D C40C               LDI     H(BEGIN)        ; IS EXECUTED IMMEDIATELY
004F CAFA               ST      PCHIGH(P2)
0051 C400   CLEAR:      LDI     0
0053 CAEA               ST      TEMP(P2)
0055 01                 XAE
0056 C400   CLEAR1:     LDI     0               ;SET ALL VARIABLES
0058 CA80               ST      EREG(P2)        ; TO ZERO
005A AAEA               ILD     TEMP(P2)
005C 01                 XAE
005D C434               LDI     52
```

```
128 005F 60             XRE
129 0060 9CF4           JNZ     CLEAR1
130 0062 C450           LDI     L(AESTK)        ;INITIALIZE SOME STACKS:
131 0064 CAFD           ST      LSTK(P2)        ; ARITHMETIC STACK,
132 0066 C47A           LDI     L(DOSTAK)
133 0068 CAFF           ST      DOPTR(P2)       ; DO/UNTIL STACK,
134 006A C46A           LDI     L(SBRSTK)
135 006C CAFC           ST      SBRPTR(P2)      ; GOSUB STACK,
136 006E C4A6           LDI     L(PCSTAK)
137 0070 CAF9           ST      PCSTK(P2)       ; I.L. CALL STACK,
138 0072 C48A           LDI     L(FORSTK)
139 0074 CAFE           ST      FORPTR(P2)      ; FOR/NEXT STACK
140
141
142             ;************************************
143             ;*   INTERMEDIATE LANGUAGE EXECUTOR  *
144             ;************************************
145
146 0076 C2FB  EXECIL:  LD      PCLOW(P2)       ;SET P3 TO CURRENT
147 0078 33             XPAL    P3              ; IL PC.
148 0079 C2FA           LD      PCHIGH(P2)
149 007B 37             XPAH    P3
150 007C C701  CHEAT:   LD      @1(P3)
151 007E 01             XAE                     ;GET NEW IL INSTRUCTION
152 007F C701           LD      @1(P3)          ; INTO P3 THROUGH
153 0081 33             XPAL    P3              ; OBSCURE METHODS
154 0082 CAFB           ST      PCLOW(P2)       ;SIMULTANEOUSLY, INCREMENT
155 0084 40             LDE                     ; THE IL PC BY 2
156 0085 37             XPAH    P3
157 0086 CAFA           ST      PCHIGH(P2)
158 0088 40             LDE
159 0089 D4F0           ANI     0F0             ;CHECK IF IL INSTRUCTION
160 008B E420           XRI     TSTBIT          ; IS A 'TEST'
161 008D 9836           JZ      TST
162 008F E4A0           XRI     CALBIT!TSTBIT   ;CHECK FOR IL CALL
163 0091 980D           JZ      ILCALL
164 0093 E4C0           XRI     JMPBIT!CALBIT   ;CHECK FOR IL JUMP
165 0095 9C06           JNZ     NOJUMP
166 0097 37             XPAH    P3              ;*** I.L. JUMP ***
167 0098 D40F           ANI     0F              ;ALL IT TAKES IS SCRUBBING
168 009A 37             XPAH    P3              ; THE JUMP FLAG OFF OF P3
169 009B 90DF  CHEAT1:  JMP     CHEAT
170 009D 3F    NOJUMP:  XPPC    P3              ;MUST BE AN ML SUBROUTINE
171 009E 90D6           JMP     EXECIL          ; IF NONE OF THE ABOVE
172
173
174             ;************************************
175             ;*      INTERMEDIATE LANGUAGE CALL    *
176             ;************************************
177
178 00A0 C2F9  ILCALL:  LD      PCSTK(P2)
179 00A2 E4D6           XRI     L(LBUF)         ;CHECK FOR STACK OVERFLOW
180 00A4 9C04           JNZ     ILC1
181 00A6 C40A           LDI     10
182 00A8 9063           JMP     E0A
183 00AA E4D6  ILC1:    XRI     L(LBUF)         ;RESTORE ACCUMULATOR
184 00AC 33             XPAL    P3              ;SAVE LOW BYTE OF NEW
185 00AD CAEA           ST      TEMP(P2)        ; I.L. PC IN TEMP
186 00AF C410           LDI     H(PCSTAK)       ;POINT P3 AT I.L.
187 00B1 37             XPAH    P3              ; SUBROUTINE STACK
188 00B2 C2FB           LD      PCLOW(P2)       ;SAVE OLD I.L. PC ON STACK
189 00B4 CF01           ST      @1(P3)
190 00B6 C2FA           LD      PCHIGH(P2)
191 00B8 CF01           ST      @1(P3)
192 00BA C2EA           LD      TEMP(P2)        ;GET LOW BYTE OF NEW
193 00BC 33             XPAL    P3              ; I.L. PC INTO P3 LOW
194 00BD CAF9           ST      PCSTK(P2)       ;UPDATE I.L. STACK POINTER
195 00BF 40             LDE                     ;GET HIGH BYTE OF NEW P3
196 00C0 D40F           ANI     0F              ; I.L. PC INTO P3 HIGH
197 00C2 37             XPAH    P3
198 00C3 90B7           JMP     CHEAT
199
200
```

```
                        ;*************************************
                        ;*       I. L.  'TEST'  INSTRUCTION      *
                        ;*************************************
                        
                                LOCAL
00C5 CAE7   TST:        ST      CHRNUM(P2)      ;CLEAR NUMBER OF CHARS SCANNE
00C7 C501   $SCAN:      LD      @1(P1)          ;SLEW OFF SPACES
00C9 E420               XRI     ' '
00CB 98FA               JZ      $SCAN
00CD C5FF               LD      @-1(P1)         ;REPOSITION CURSOR
00CF C2FA               LD      PCHIGH(P2)      ;POINT P3 AT IL TABLE
00D1 37                 XPAH    P3
00D2 D40F               ANI     0F              ;FAIL ADDRESS <- OLD P3
00D4 CAEC               ST      FAILHI(P2)
00D6 C2FB               LD      PCLOW(P2)
00D8 33                 XPAL    P3
00D9 CAED               ST      FAILLO(P2)
00DB C701   $LOOP:      LD      @1(P3)
00DD 01                 XAE                     ;SAVE CHAR FROM TABLE
00DE BAE7               DLD     CHRNUM(P2)      ;DECREMENT CHAR COUNT
00E0 40                 LDE                     ;GET CHAR BACK
00E1 D47F               ANI     07F             ;SCRUB OFF FLAG (IF ANY)
00E3 E501               XOR     @1(P1)          ;IS CHAR EQUAL TO TEXT CHAR?
00E5 9C07               JNZ     $NEQ            ;NO - END TEST
00E7 40                 LDE                     ;YES - BUT IS IT LAST CHAR?
00E8 94F1               JP      $LOOP           ;IF NOT, CONTINUE TO COMPARE
00EA 9090               JMP     CHEAT           ;IF SO, GET NEXT I.L.
00EC 9088   XO:         JMP     EXECIL          ; INSTRUCTION
00EE C2E7   $NEQ:       LD      CHRNUM(P2)      ;RESTORE P1 TO
00F0 01                 XAE                     ; ORIGINAL VALUE
00F1 C580               LD      @EREG(P1)
00F3 C2ED               LD      FAILLO(P2)      ;LOAD TEST-FAIL ADDRESS
00F5 33                 XPAL    P3              ; INTO P3
00F6 C2EC               LD      FAILHI(P2)
00F8 37                 XPAH    P3
00F9 90A0               JMP     CHEAT1          ;GET NEXT IL INSTRUCTION
                        
                        
                        ;*************************************
                        ;*        I. L.  SUBROUTINE RETURN     *
                        ;*************************************
                        
00FB C410   RTN:        LDI     H(PCSTAK)       ;POINT P3 AT I. L.  PC STACK
00FD 37                 XPAH    P3
00FE C2F9               LD      PCSTK(P2)
0100 33                 XPAL    P3
0101 C7FF               LD      @-1(P3)         ;GET HIGH PART OF OLD PC
0103 01                 XAE
0104 C7FF               LD      @-1(P3)         ;GET LOW PART OF OLD PC
0106 33                 XPAL    P3
0107 CAF9               ST      PCSTK(P2)       ;UPDATE IL STACK POINTER
0109 40                 LDE
010A 37                 XPAH    P3              ;P3 NOW HAS OLD IL PC
010B 908E               JMP     CHEAT1
010D 9041   EOA:        JMP     E0
                        
                        
                        ;*************************************
                        ;*     SAVE GOSUB RETURN ADDRESS     *
                        ;*************************************
                        
010F C2FC   SAV:        LD      SBRPTR(P2)
0111 E47A               XRI     L(DOSTAK)       ;CHECK FOR MORE
0113 981C               JZ      SAV2            ; THAN 8 SAVES
0115 AAFC               ILD     SBRPTR(P2)
0117 AAFC               ILD     SBRPTR(P2)
0119 33                 XPAL    P3              ;SET P3 TO
011A C410               LDI     H(SBRSTK)       ; SUBROUTINE STACK TOP.
011C 37                 XPAH    P3
011D C2F4               LD      RUNMOD(P2)      ;IF IMMEDIATE MODE,
011F 980A               JZ      SAV1            ; SAVE NEGATIVE ADDRESS.
0121 35                 XPAH    P1              ;SAVE HIGH PORTION
```

```
0122 CBFF            ST      -1(P3)          ; OF CURSOR
0124 35              XPAH    P1
0125 31              XPAL    P1              ;SAVE LOW PORTION
0126 CBFE            ST      -2(P3)          ; OF CURSOR
0128 31              XPAL    P1
0129 90C1            JMP     XQ              ;RETURN
012B C4FF    SAV1    LDI     -1              ;IMMEDIATE MODE
012D CBFF            ST      -1(P3)          ; RETURN ADDRESS IS
012F 90BB            JMP     XQ              ; NEGATIVE.
0131 C40A    SAV2    LDI     10              ;ERROR: MORE THAN
0133 901B            JMP     EQ              ; 8 GOSUBS

                     ;*************************************
                     ;*      CHECK STATEMENT FINISHED       *
                     ;*************************************

0135 C501    DONE    LD      @1(P1)          ;SKIP SPACES
0137 E420            XRI     ' '
0139 98FA            JZ      DONE
013B E42D            XRI     '-' OD          ;IS IT CARRIAGE RETURN?
013D 9804            JZ      DONE1           ;YES - RETURN
013F E437            XRI     037             ;IS CHAR A '?' ?
0141 9C01            JNZ     DONE2           ;NO - ERROR
0143 3F      DONE1   XPPC    P3              ;YES - RETURN
0144 C404    DONE2:  LDI     4
0146 9008            JMP     EQ

                     ;*************************************
                     ;*      RETURN FROM GOSUB              *
                     ;*************************************

0148 C2FC    RSTR    LD      SBRPTR(P2)
014A E46A            XRI     L(SBRSTK)       ;CHECK FOR RETURN
014C 9C04            JNZ     RSTR1           ; W/O GOSUB.
014E C409            LDI     9
0150 9043    EQ      JMP     E1              ;GOTO ERROR.
0152 BAFC    RSTR1   DLD     SBRPTR(P2)
0154 BAFC            DLD     SBRPTR(P2)      ;POP GOSUB STACK,
0156 33              XPAL    P3              ; PUT PTR INTO P3.
0157 C410            LDI     H(SBRSTK)
0159 37              XPAH    P3
015A C301            LD      1(P3)           ;IF ADDRESS NEGATIVE,
015C 9409            JP      RSTR2           ; SUBROUTINE WAS CALLED
015E C402            JS      P3,FIN          ; IN IMMEDIATE MODE,
0160 9085    X1      JMP     XQ              ; SO FINISH UP EXECUTING
0162 35      RSTR2:  XPAH    P1              ;RESTORE CURSOR HIGH
0163 C300            LD      0(P3)
0165 31              XPAL    P1              ;RESTORE CURSOR LOW
0166 C401            LDI     1               ;SET RUN MODE
0168 CAF4            ST      RUNMOD(P2)
016A 90F4            JMP     X1

                     ;*************************************
                     ;*      TRANSFER TO NEW STATEMENT      *
                     ;*************************************

016C C2F2    XFER:   LD      LABLHI(P2)      ;CHECK FOR NON-EXISTENT LINE
016E 9404            JP      XFER1
0170 C408            LDI     8
0172 901C            JMP     E1
0174 C401    XFER1   LDI     1               ;SET RUN MODE TO 1
0176 CAF4            ST      RUNMOD(P2)
0178 3F              XPPC    P3

                     ;*************************************
                     ;*      PRINT STRING IN TEXT           *
                     ;*************************************

017A         PRS     LDPI    P3,PUTC-1       ;POINT P3 AT PUTC ROUTINE
```

```
0184 C501           LD      @1(P1)          ;LOAD NEXT CHAR
0186 E422           XRI     '"'             ;IF ", END OF
0188 98DB           JZ      X1              ; STRING
018A E42F           XRI     02F             ;IF CR, ERROR
018C 9805           JZ      PRS1
018E E40D           XRI     0D              ;RESTORE CHAR
0190 3F             XPPC    P3              ;PRINT CHAR
0191 90EB           JMP     PRS             ;GET NEXT CHAR
0193 C407   PRS1    LDI     7               ;SYNTAX ERROR
0195 9035   E1      JMP     E2
                    ;*************************************
                    ;*      PRINT NUMBER ON STACK        *
                    ;*************************************
                    ; THIS ROUTINE IS BASED ON DENNIS ALLISON'S BINARY TO DECIMAL
                    ; CONVERSION ROUTINE IN VOL. 1, #1 OF "DR. DOBB'S JOURNAL",
                    ; BUT IS MUCH MORE OBSCURE BECAUSE OF THE STACK MANIPULATION.
                        .LOCAL
0197 C410   PRN     LDI     H(AESTK)        ;POINT P3 AT A.E. STACK
0199 37             XPAH    P3
019A AAFD           ILD     LSTK(P2)
019C AAFD           ILD     LSTK(P2)
019E 33             XPAL    P3
019F C40A           LDI     10              ;PUT 10 ON STACK (WE'LL BE
01A1 CBFE           ST      -2(P3)          ; DIVIDING BY IT LATER)
01A3 C400           LDI     0
01A5 CBFF           ST      -1(P3)
01A7 C405           LDI     5               ;SET CHRNUM TO POINT TO PLACE
01A9 CAE7           ST      CHRNUM(P2)      ; IN STACK WHERE WE STORE
01AB C4FF           LDI     -1              ; THE CHARACTERS TO PRINT
01AD CB05           ST      5(P3)           ;FIRST CHAR IS A FLAG (-1)
01AF C3FD           LD      -3(P3)          ;CHECK IF NUMBER IS NEGATIVE
01B1 9413           JP      $1
01B3 C42D           LDI     '-'             ;PUT '-' ON STACK, AND NEGATE
01B5 CB04           ST      4(P3)           ; THE NUMBER
01B7 C400           LDI     0
01B9 03             SCL
01BA FBFC           CAD     -4(P3)
01BC CBFC           ST      -4(P3)
01BE C400           LDI     0
01C0 FBFD           CAD     -3(P3)
01C2 CBFD           ST      -3(P3)
01C4 909F           JMP     X1              ;GO DO DIVISION BY 10
01C6 C420   $1      LDI     ' '             ;IF POSITIVE, PUT ' ' ON
01C8 CB04           ST      4(P3)           ; STACK BEFORE DIVISION
01CA 9099   X4      JMP     X1
01CC 9057   E2      JMP     ERR1
                    ; THE DIVISION IS PERFORMED, THEN CONTROL IS TRANSFERRED
                    ; TO PRN1, WHICH FOLLOWS.
01CE AAFD   PRN1    ILD     LSTK(P2)        ;POINT P1 AT A.E. STACK
01D0 AAFD           ILD     LSTK(P2)
01D2 31             XPAL    P1
01D3 C410           LDI     H(AESTK)
01D5 35             XPAH    P1
01D6 AAE7           ILD     CHRNUM(P2)      ;INCREMENT CHARACTER STACK
01D8 01             XAE                     ; POINTER, PUT IN EX. REG.
01D9 C101           LD      1(P1)           ;GET REMAINDER FROM DIVIDE,
01DB DC30           ORI     '0'
01DD C980           ST      EREG(P1)        ; PUT IT ON THE STACK
01DF C1FD           LD      -3(P1)          ;IS THE QUOTIENT ZERO YET?
01E1 D9FC           OR      -4(P1)
01E3 980A           JZ      $PRNT           ;YES - GO PRINT THE NUMBER
01E5 C40F           LDI     H(PRNUM1)       ;NO - CHANGE THE I.L. PC
01E7 CAFA           ST      PCHIGH(P2)      ; SO THAT DIVIDE IS
01E9 C42F           LDI     L(PRNUM1)       ; PERFORMED AGAIN
01EB CAFB           ST      PCLOW(P2)
01ED 90DB           JMP     X4              ;GO DO DIVISION BY 10 AGAIN
01EF        $PRNT   LDPI    P3,PUTC-1       ;POINT P3 AT PUTC ROUTINE
01F5 C2F5           LD      LISTNG(P2)      ;IF LISTING, SKIP PRINTING
```

```
420 01F7 9C06           JNZ     $2              ; LEADING SPACE
421 01F9 C104           LD      4(P1)           ;PRINT EITHER '-'
422 01FB 3F             XPPC    P3              ; OR LEADING SPACE
423 01FC C2E7           LD      CHRNUM(P2)      ;GET EX. REG. VALUE BACK
424 01FE 01             XAE
425 01FF C580   $2      LD      @EREG(P1)       ;POINT P3 AT FIRST CHAR
426 0201 C100           LD      (P1)            ; TO BE PRINTED
427 0203 3F     $LOOP   XPPC    P3              ;PRINT THE CHARACTER
428 0204 C5FF           LD      @-1(P1)         ;GET NEXT CHARACTER
429 0206 94FB           JP      $LOOP           ;REPEAT UNTIL = -1
430 0208 C450           LDI     L(AESTK)
431 020A CAFD           ST      LSTK(P2)        ;CLEAR THE A.E. STACK
432 020C C2F5           LD      LISTNG(P2)      ;PRINT A TRAILING SPACE

433 020E 9CBA           JNZ     X4              ; IF NOT LISTING PROGRAM
434 0210 C420           LDI     ' '
435 0212 3F             XPPC    P3
436 0213 90B5           JMP     X4
437
438
439             ;************************************
440             ;*      CARRIAGE RETURN/LINE FEED     *
441             ;************************************
442
443 0215        NLINE   LDPI    P3,PUTC-1       ;POINT P3 AT PUTC ROUTINE
444 0217 C40D           LDI     0D              ;CARRIAGE RETURN
445 0219 3F             XPPC    P3
446 021A C40A           LDI     0A              ;LINE FEED
447 021C 3F             XPPC    P3
448 021D 90A7   X5      JMP     X4
449
450
451             ;************************************
452             ;*      ERROR  ROUTINE              *
453             ;************************************
454
455                     LOCAL
456 021F C405   ERR     LDI     5               ;SYNTAX ERROR
457 0221 CAEB   ERR1    ST      NUM(P2)         ;SAVE ERROR #
458 0223 C2EB   ERR2    LD      NUM(P2)
459 0225 CAEA           ST      TEMP(P2)
460 0227                LDPI    P3,PUTC-1       ;POINT P3 AT PUTC
461 0229 C40D           LDI     0D              ;PRINT CR/LF
462 022B 3F             XPPC    P3
463 022C C40A           LDI     0A
464 022E 3F             XPPC    P3
465 022F                LDPI    P1,MESGS        ;P1 -> ERROR MESSAGES
466 0231 BAEB   $1      DLD     NUM(P2)         ;IS THIS THE RIGHT MESSAGE?
467 0233 9806           JZ      $MSG            ;YES - GO PRINT IT
468 0235 C501   $LOOP   LD      @1(P1)          ;NO - SCAN THROUGH TO
469 0237 94FC           JP      $LOOP           ; NEXT MESSAGE
470 0239 90F6           JMP     $1
471 023B C501   $MSG    LD      @1(P1)          ;GET MESSAGE CHAR
472 023D 3F             XPPC    P3              ;PRINT IT
473 023E C1FF           LD      -1(P1)          ;IS MESSAGE DONE?
474 0240 94F9           JP      $MSG            ;NO - GET NEXT CHAR
475 0242 C2EA           LD      TEMP(P2)        ;WAS THIS A BREAK MESSAGE?
476 0244 E40E           XRI     14
477 0246 980D           JZ      $3              ;YES - SKIP PRINTING 'ERROR'
478 0248                LDPI    P1,MESGS        ;NO - PRINT 'ERROR'
479 024A C501   $2      LD      @1(P1)          ;GET CHARACTER
480 024C 3F             XPPC    P3              ;PRINT IT
481 024D C1FF           LD      -1(P1)          ;DONE?
482 024F 94F9           JP      $2              ;NO - REPEAT LOOP
483 0251 C2F4   $3      LD      RUNMOD(P2)      ;DON'T PRINT LINE #
484 0253 984D           JZ      FIN             ; IF IMMEDIATE MODE
485 0255 C420           LDI     ' '
486 0257 3F             XPPC    P3              ;SPACE

487 0258 C441           LDI     'A'             ;AT
488 025A 3F             XPPC    P3
489 025B C454           LDI     'T'
490 025D 3F             XPPC    P3
491 025E C410           LDI     H(AESTK)        ;POINT P3 AT A.E. STACK
492 0260 37             XPAH    P3
```

```
493 0271 AAFD            ILD     LSTK(P2)
494 0273 AAFD            ILD     LSTK(P2)
495 0275 33              XPAL    P3
496 0276 C2F7            LD      HILINE(P2)      ;GET HIGH BYTE OF LINE #
497 0278 CBFF            ST      -1(P3)          ;PUT ON STACK
498 027A C2F8            LD      LOLINE(P2)      ;GET LOW BYTE OF LINE #
499 027C CBFE            ST      -2(P3)          ;PUT ON STACK
500 027E C42D            LDI     L(ERRNUM)       ;GO TO PRN
501 0280 CAFB            ST      PCLOW(P2)
502 0282 C40E            LDI     H(ERRNUM)
503 0284 CAFA            ST      PCHIGH(P2)
504 0286 9099    X5A:    JMP     X5
505
506
507              ;************************************
508              ;*      BREAK, NXT, FIN, & STRT     *
509              ;************************************
510
511 0288 C40E    BREAK:  LDI     14
512 028A 9099    E3A:    JMP     ERR1
513                                              ;*** NEXT STATEMENT ***
514 028C C2F4    NXT:    LD      RUNMOD(P2)      ;IF IN IMMED. MODE,
515 028E 9822            JZ      FIN             ; STOP EXECUTION
516 0290 C100            LD      (P1)            ;IF WE HIT END OF FILE,
517 0292 D480            ANI     080             ; FINISH UP THINGS
518 0294 9C1C            JNZ     FIN
519 0296 06              CSA                     ;BREAK IF SOMEONE IS
520 0297 D420            ANI     020             ; TYPING ON THE CONSOLE
521 0299 98ED            JZ      BREAK
522 029B C1FF            LD      -1(P1)          ;GET LAST CHARACTER SCANNED
523 029D E40D            XRI     0D              ;WAS IT CARRIAGE RETURN?
524 029F 9C08            JNZ     NXT1            ;YES - SKIP FOLLOWING UPDATES
525 02A1 C501            LD      @1(P1)          ;GET HIGH BYTE OF NEXT LINE #
526 02A3 CAF7            ST      HILINE(P2)      ;SAVE IT
527 02A5 C502            LD      @2(P1)          ;GET LOW BYTE OF LINE #, SKIP
528 02A7 CAF8            ST      LOLINE(P2)      ; LINE LENGTH BYTE
529 02A9 C40C    NXT1:   LDI     H(STMT)         ;GO TO 'STMT' IN IL TABLE
530 02AB CAFA            ST      PCHIGH(P2)
531 02AD C482            LDI     L(STMT)
532 02AF CAFB            ST      PCLOW(P2)
533 02B1 3F              XPPC    P3
534
535 02B2 C400    FIN:    LDI     0               ;*** FINISH EXECUTION ***
536 02B4 CAF4            ST      RUNMOD(P2)      ;CLEAR RUN MODE
537 02B6 C450            LDI     L(AESTK)        ;CLEAR ARITHMETIC STACK
538 02B8 CAFD            ST      LSTK(P2)
539 02BA C418            LDI     L(START)        ;SET IL PC TO GETTING LINES
540 02BC CAFB            ST      PCLOW(P2)
541 02BE C40C            LDI     H(START)
542 02C0 CAFA            ST      PCHIGH(P2)
543 02C2 C4A6            LDI     L(PCSTAK)
544 02C4 CAF9            ST      PCSTK(P2)
545 02C6 90BE            JMP     X5A
546                                              ;*** START EXECUTION ***
547 02C8 AAF4    STRT:   ILD     RUNMOD(P2)      ;RUN MODE = 1
548 02CA C2E9            LD      TEMP2(P2)       ;POINT CURSOR TO
549 02CC 35              XPAH    P1              ; START OF NIBL PROGRAM
550 02CD C2E8            LD      TEMP3(P2)
551 02CF 31              XPAL    P1
552 02D0 C46A            LDI     L(SBRSTK)       ;EMPTY SOME STACKS:
553 02D2 CAFC            ST      SBRPTR(P2)      ; GOSUB STACK,
554 02D4 C48A            LDI     L(FORSTK)
555 02D6 CAFE            ST      FORPTR(P2)      ; FOR STACK
556 02D8 C47A            LDI     L(DOSTAK)
557 02DA CAFF            ST      DOPTR(P2)       ; & DO/UNTIL STACK
558 02DC 3F              XPPC    P3              ;RETURN
559 02DD 90A7    X6:     JMP     X5A
560 02DF 90A9    E4:     JMP     E3A
561
562
563              ;************************************
564              ;*        LIST NIBL PROGRAM         *
565              ;************************************
```

```
566
567 02E1 C100   LST     LD      (P1)            ;CHECK FOR END OF FILE
568 02E3 E480           XRI     080
569 02E5 9418           JP      LST2
570 02E7 C410           LDI     H(AESTK)        ;GET LINE NUMBER ONTO STACK
571 02E9 37             XPAH    P3
572 02EA AAFD           ILD     LSTK(P2)
573 02EC AAFD           ILD     LSTK(P2)
574 02EE 33             XPAL    P3
575 02EF C501           LD      @1(P1)
576 02F1 CBFF           ST      -1(P3)
577 02F3 C501           LD      @1(P1)
578 02F5 CBFE           ST      -2(P3)
579 02F7 C501           LD      @1(P1)          ;SKIP OVER LINE LENGTH
580 02F9 C401           LDI     1
581 02FB CAF5           ST      LISTNG(P2)      ;SET LISTING FLAG
582 02FD 90DE           JMP     X6              ;GO PRINT LINE NUMBER
583 02FF C400   LST2    LDI     0
584 0301 CAF5           ST      LISTNG(P2)      ;CLEAR LISTING FLAG
585 0303 C402           JS      P3,NXT          ;GO TO NXT
586 030A 90D1   X6A     JMP     X6
587 030C 90D1   E5      JMP     E4
588 030E        LST3    LDPI    P3,PUTC-1       ;POINT P3 AT PUTC
589 0314 06     LST4    CSA
590 0315 D420           ANI     020
591 0317 98E6           JZ      LST2            ;IF TYPING, STOP
592 0319 C501           LD      @1(P1)          ;GET NEXT CHAR
593 031B E40D           XRI     0D              ;TEST FOR CR
594 031D 9805           JZ      LST5
595 031F E40D           XRI     0D              ;GET CHARACTER
596 0321 3F             XPPC    P3              ;PRINT CHARACTER
597 0322 90F0           JMP     LST4
598 0324 C40D   LST5    LDI     0D              ;CARRIAGE RETURN
599 0326 3F             XPPC    P3
600 0327 C40A           LDI     0A              ;LINE FEED
601 0329 3F             XPPC    P3
602 032A 02             CCL
603 032B C447           LDI     L(LIST3)
604 032D CAFB           ST      PCLOW(P2)
605 032F C400           LDI     H(LIST3)
606 0331 CAFA           ST      PCHIGH(P2)
607 0333 90AC           JMP     LST             ;GET NEXT LINE
608
609
610             ;************************************
611             ;*          ADD AND SUBTRACT        *
612             ;************************************
613
614 0335 C410   ADD     LDI     H(AESTK)        ;SET P3 TO CURRENT
615 0337 37             XPAH    P3              ; STACK LOCATION
616 0338 BAFD           DLD     LSTK(P2)
617 033A BAFD           DLD     LSTK(P2)
618 033C 33             XPAL    P3
619 033D 02             CCL
620 033E C3FE           LD      -2(P3)          ;REPLACE TWO TOP ITEMS
621 0340 F300           ADD     0(P3)           ; ON STACK BY THEIR SUM
622 0342 CBFE           ST      -2(P3)
623 0344 C3FF           LD      -1(P3)
624 0346 F301           ADD     1(P3)
625 0348 CBFF           ST      -1(P3)
626 034A 90BE   X7      JMP     X6A
627
628 034C C410   SUB     LDI     H(AESTK)        ;SET P3 TO CURRENT
629 034E 37             XPAH    P3              ; STACK LOCATION
630 034F BAFD           DLD     LSTK(P2)
631 0351 BAFD           DLD     LSTK(P2)
632 0353 33             XPAL    P3
633 0354 03             SCL
634 0355 C3FE           LD      -2(P3)          ;REPLACE TWO TOP ITEMS
635 0357 FB00           CAD     0(P3)           ; ON STACK BY THEIR DIFFERENC
636 0359 CBFE           ST      -2(P3)
637 035B C3FF           LD      -1(P3)
638 035D FB01           CAD     1(P3)
```

```
0361 90A7           JMP     X6A
                    ;*************************************
                    ;*          NEGATE                   *
                    ;*************************************
0363 C410   NEG:    LDI     H(AESTK)        ;SET P3 TO CURRENT
0365 37             XPAH    P3              ; STACK LOCATION
0366 C2FD           LD      LSTK(P2)
0368 33             XPAL    P3
0369 03             SCL
036A C400           LDI     0
036C FBFE           CAD     -2(P3)          ;NEGATE TOP ITEM ON STACK
036E CBFE           ST      -2(P3)
0370 C400           LDI     0
0372 FBFF           CAD     -1(P3)
0374 CBFF           ST      -1(P3)
0376 90D2   X8:     JMP     X7
0378 9092   E6:     JMP     E5
                    ;*************************************
                    ;*          MULTIPLY                 *
                    ;*************************************
                          LOCAL
037A C410   MUL:    LDI     H(AESTK)        ;SET P3 TO CURRENT
037C 37             XPAH    P3              ; STACK LOCATION
037D C2FD           LD      LSTK(P2)
037F 33             XPAL    P3              ;DETERMINE SIGN OF PRODUCT,
0380 C3FF           LD      -1(P3)          ; SAVE IN TEMP(P2)
0382 E3FD           XOR     -3(P3)
0384 CAEA           ST      TEMP(P2)
0386 C3FF           LD      -1(P3)          ;CHECK FOR NEGATIVE
0388 940D           JP      $1              ; MULTIPLIER
038A 03             SCL
038B C400           LDI     0               ;IF NEGATIVE,
038D FBFE           CAD     -2(P3)          ; NEGATE
038F CBFE           ST      -2(P3)
0391 C400           LDI     0
0393 FBFF           CAD     -1(P3)
0395 CBFF           ST      -1(P3)
0397 C3FD   $1:     LD      -3(P3)          ;CHECK FOR NEGATIVE
0399 940D           JP      $2              ; MULTIPLICAND
039B 03             SCL
039C C400           LDI     0               ;IF NEGATIVE,
039E FBFC           CAD     -4(P3)          ; NEGATE
03A0 CBFC           ST      -4(P3)
03A2 C400           LDI     0
03A4 FBFD           CAD     -3(P3)
03A6 CBFD           ST      -3(P3)
03A8 C400   $2      LDI     0               ;CLEAR WORKSPACE
03AA CB00           ST      0(P3)
03AC CB01           ST      1(P3)
03AE CB02           ST      2(P3)
03B0 CB03           ST      3(P3)
03B2 C410           LDI     16              ;SET COUNTER TO 16
03B4 CAEB           ST      NUM(P2)
03B6 C3FF   $LOOP   LD      -1(P3)          ;ROTATE MULTIPLIER
03B8 1F             RRL                     ; RIGHT ONE BIT
03B9 CBFF           ST      -1(P3)
03BB C3FE           LD      -2(P3)
03BD 1F             RRL
03BE CBFE           ST      -2(P3)
03C0 06             CSA                     ;CHECK FOR CARRY BIT
03C1 9411           JP      $3              ;IF NOT SET, DON'T DO ADD
03C3 02             CCL
03C4 C302           LD      2(P3)           ;ADD MULTIPLICAND
03C6 F3FC           ADD     -4(P3)          ; INTO WORKSPACE
03C8 CB02           ST      2(P3)
```

```
711 03CA C303           LD      3(P3)
712 03CC F3FD           ADD     -3(P3)
713 03CE CB03           ST      3(P3)
714 03D0 9002           JMP     $3
715 03D2 90A4   E6A     JMP     E6
716 03D4 02     $3      CCL
717 03D5 C303           LD      3(P3)           ;SHIFT WORKSPACE RIGHT BY 1
718 03D7 1F             RRL
719 03D8 CB03           ST      3(P3)
720 03DA C302           LD      2(P3)
721 03DC 1F             RRL
722 03DD CB02           ST      2(P3)
723 03DF C301           LD      1(P3)
724 03E1 1F             RRL
725 03E2 CB01           ST      1(P3)
726 03E4 C300           LD      0(P3)
727 03E6 1F             RRL
728 03E7 CB00           ST      0(P3)
729 03E9 BAEB           DLD     NUM(P2)         ;DECREMENT COUNTER
730 03EB 9CC9           JNZ     $LOOP           ;LOOP IF NOT ZERO
731 03ED 9002           JMP     $4
732 03EF 9085   X9      JMP     X8
733 03F1 C2EA   $4      LD      TEMP(P2)        ;CHECK SIGN WORD
734 03F3 940D           JP      $EXIT           ;IF BIT7 = 1, NEGATE PRODUCT
735 03F5 03             SCL
736 03F6 C400           LDI     0
737 03F8 FB00           CAD     0(P3)
738 03FA CB00           ST      0(P3)
739 03FC C400           LDI     0
740 03FE FB01           CAD     1(P3)
741 0400 CB01           ST      1(P3)
742 0402 C300   $EXIT   LD      0(P3)           ;PUT PRODUCT ON TOP
743 0404 CBFC           ST      -4(P3)          ; OF STACK
744 0406 C301           LD      1(P3)
745 0408 CBFD           ST      -3(P3)
746 040A BAFD           DLD     LSTK(P2)        ;SUBTRACT 2 FROM
747 040C BAFD           DLD     LSTK(P2)        ; LSTK
748 040E 90DF           JMP     X9
749
750
751             ;*************************************
752             ;*              DIVIDE               *
753             ;*************************************
754
755                     .LOCAL
756 0410 C410   DIV     LDI     H(AESTK)
757 0412 37             XPAH    P3
758 0413 C2FD           LD      LSTK(P2)
759 0415 33             XPAL    P3
760 0416 C3FF           LD      -1(P3)          ;CHECK FOR DIVISION BY 0
761 0418 DBFE           OR      -2(P3)
762 041A 9C04           JNZ     $0
763 041C C40D           LDI     13
764 041E 90B2           JMP     E6A
765 0420 C3FD   $0      LD      -3(P3)
766 0422 E3FF           XOR     -1(P3)
767 0424 CAEA           ST      TEMP(P2)        ;SAVE SIGN OF QUOTIENT
768 0426 C3FD           LD      -3(P3)          ;IS DIVIDEND POSITIVE?
769 0428 9411           JP      $POS            ;YES - JUMP
770 042A C400           LDI     0
771 042C 03             SCL
772 042D FBFC           CAD     -4(P3)          ;NO - NEGATE DIVIDEND,
773 042F CB03           ST      3(P3)           ; STORE IN RIGHT HALF
774 0431 C400           LDI     0               ; OF 32-BIT ACCUMULATOR
775 0433 FBFD           CAD     -3(P3)
776 0435 CB02           ST      2(P3)
777 0437 900A           JMP     $1
778 0439 90B4   X9A     JMP     X9
779 043B C3FD   $POS    LD      -3(P3)          ;STORE NON-NEGATED DIVIDEND
780 043D CB02           ST      2(P3)           ; IN 32-BIT ACCUMULATOR
781 043F C3FC           LD      -4(P3)
782 0441 CB03           ST      3(P3)
783 0443 C3FF   $1      LD      -1(P3)          ;CHECK FOR NEGATIVE DIVISOR
```

```
784 0445 940D           JP      $2
785 0447 C400           LDI     0               ;NEGATE DIVISOR
786 0449 03             SCL
787 044A FBFE           CAD     -2(P3)
788 044C CBFE           ST      -2(P3)
789 044E C400           LDI     0
790 0450 FBFF           CAD     -1(P3)
791 0452 CBFF           ST      -1(P3)
792 0454 C400   $2      LDI     0               ;PUT ZERO IN
793 0456 CB01           ST      1(P3)           ;  LEFT HALF OF 32-BIT ACC,
794 0458 CB00           ST      0(P3)
795 045A CAEB           ST      NUM(P2)         ;  THE COUNTER, AND
796 045C CBFD           ST      -3(P3)          ;  IN THE DIVIDEND, NOW USED
797 045E CBFC           ST      -4(P3)          ;  STORE THE QUOTIENT
798 0460 02     $LOOP   CCL                     ;BEGIN MAIN DIVIDE LOOP:
799 0461 C3FC           LD      -4(P3)          ;  SHIFT QUOTIENT LEFT,
800 0463 F3FC           ADD     -4(P3)
801 0465 CBFC           ST      -4(P3)
802 0467 C3FD           LD      -3(P3)
803 0469 F3FD           ADD     -3(P3)
804 046B CBFD           ST      -3(P3)
805 046D 02             CCL                     ;  SHIFT 32-BIT ACC LEFT,
806 046E C303           LD      3(P3)
807 0470 F303           ADD     3(P3)
808 0472 CB03           ST      3(P3)
809 0474 C302           LD      2(P3)
810 0476 F302           ADD     2(P3)
811 0478 CB02           ST      2(P3)
812 047A C301           LD      1(P3)
813 047C F301           ADD     1(P3)
814 047E CB01           ST      1(P3)
815 0480 C300           LD      (P3)
816 0482 F300           ADD     (P3)
817 0484 CB00           ST      (P3)
818 0486 03             SCL
819 0487 C301           LD      1(P3)           ;  SUBTRACT DIVISOR INTO
820 0489 FBFE           CAD     -2(P3)          ;   LEFT HALF OF ACC,
821 048B CB01           ST      1(P3)
822 048D C300           LD      (P3)
823 048F FBFF           CAD     -1(P3)
824 0491 CB00           ST      (P3)
825 0493 9411           JP      $ENT1           ;  IF RESULT IS NEGATIVE,
826 0495 02             CCL                     ;   RESTORE ORIGINAL CONTENTS
827 0496 C301           LD      1(P3)           ;   OF ACC BY ADDING DIVISOR
828 0498 F3FE           ADD     -2(P3)
829 049A CB01           ST      1(P3)
830 049C C300           LD      (P3)
831 049E F3FF           ADD     -1(P3)
832 04A0 CB00           ST      (P3)
833 04A2 9008           JMP     $3
834 04A4 9093   X9B:    JMP     X9A
835 04A6 C3FC   $ENT1   LD      -4(P3)          ;  ELSE IF RESULT POSITIVE,
836 04A8 D401           ORI     1               ;   RECORD A 1 IN QUOTIENT
837 04AA CBFC           ST      -4(P3)          ;   W/O RESTORING THE ACC
838 04AC AAEB   $3      ILD     NUM(P2)         ;  INCREMENT THE COUNTER
839 04AE E410           XRI     16              ;  ARE WE DONE?
840 04B0 9CAE           JNZ     $LOOP           ;  LOOP IF NOT DONE
841 04B2 C2EA           LD      TEMP(P2)        ;CHECK THE QUOTIENT'S SIGN,
842 04B4 940D           JP      $END            ; NEGATING IF, NECESSARY
843 04B6 C400           LDI     0
844 04B8 03             SCL
845 04B9 FBFC           CAD     -4(P3)
846 04BB CBFC           ST      -4(P3)
847 04BD C400           LDI     0
848 04BF FBFD           CAD     -3(P3)
849 04C1 CBFD           ST      -3(P3)
850 04C3 BAFD   $END    DLD     LSTK(P2)        ;DECREMENT THE STACK POINTER,
851 04C5 BAFD           DLD     LSTK(P2)
852 04C7 90DB           JMP     X9B             ; AND EXIT
853
854
855             ;*************************************
856             ;*          STORE VARIABLE           *
857             ;*************************************
```

```
858
859 04C9 C410   STORE   LDI     H(AESTK)        ;SET P3 TO STACK
860 04CB 37             XPAH    P3
861 04CC C2FD           LD      LSTK(P2)
862 04CE 33             XPAL    P3
863 04CF C7FD           LD      @-3(P3)         ;GET VARIABLE INDEX
864 04D1 01             XAE                     ;PUT IN E REG
865 04D2 C301           LD      1(P3)
866 04D4 CA80           ST      EREG(P2)        ;STORE LOWER 8 BITS
867 04D6 02             CCL                     ; INTO VARIABLE
868 04D7 40             LDE                     ;INCREMENT INDEX
869 04D8 F401           ADI     1
870 04DA 01             XAE
871 04DB C302           LD      2(P3)
872 04DD CA80           ST      EREG(P2)        ;STORE UPPER 8 BITS
873 04DF 33             XPAL    P3              ; INTO VARIABLE
874 04E0 CAFD           ST      LSTK(P2)        ;UPDATE STACK POINTER
875 04E2 C400   X10:    JS      P3,EXECIL
876
877
878             ;************************************
879             ;*      TEST FOR VARIABLE IN TEXT     *
880             ;************************************
881
882 04E9 C501   TSTVAR: LD      @1(P1)
883 04EB E420           XRI     ' '             ;SLEW OFF SPACES
884 04ED 98FA           JZ      TSTVAR
885 04EF C1FF           LD      -1(P1)          ;GET CHARACTER IN QUESTION
886 04F1 03             SCL
887 04F2 FC5B           CAI     'Z'+1           ;SUBTRACT 'Z'+1
888 04F4 9405           JP      $FAIL           ;NOT VARIABLE IF POSITIVE
889 04F6 03             SCL
890 04F7 FCE6           CAI     'A'-'Z'-1       ;SUBTRACT 'A'
891 04F9 9412           JP      $MAYBE          ;IF POS, MAY BE VARIABLE
892 04FB C5FF   $FAIL:  LD      @-1(P1)         ;BACKSPACE CURSOR
893 04FD C2FB           LD      PCLOW(P2)       ;GET TEST-FAIL ADDRESS
894 04FF 33             XPAL    P3              ; FROM I.L. TABLE, PUT IT
895 0500 C2FA           LD      PCHIGH(P2)      ; INTO I.L. PROGRAM COUNTER
896 0502 37             XPAH    P3
897 0503 C300           LD      (P3)
898 0505 CAFA           ST      PCHIGH(P2)
899 0507 C301           LD      1(P3)
900 0509 CAFB           ST      PCLOW(P2)
901 050B 90D5           JMP     X10
902 050D 01     $MAYBE: XAE                     ;SAVE VALUE (0-25)
903 050E C100           LD      (P1)            ;CHECK FOLLOWING CHAR
904 0510 03             SCL                     ;MUST NOT BE A LETTER
905 0511 FC5B           CAI     'Z'+1           ; OTHERWISE WE'D BE LOOKING
906 0513 9405           JP      $OK             ; AT A KEYWORD, NOT VARIABLE
907 0515 03             SCL
908 0516 FCE6           CAI     'A'-'Z'-1
909 0518 94E1           JP      $FAIL
910 051A C410   $OK:    LDI     H(AESTK)        ;SET P3 TO CURRENT
911 051C 37             XPAH    P3              ; STACK LOCATION
912 051D AAFD           ILD     LSTK(P2)        ;INCR STACK POINTER
913 051F 33             XPAL    P3
914 0520 02             CCL                     ;DOUBLE VARIABLE INDEX
915 0521 40             LDE
916 0522 70             ADE
917 0523 CBFF           ST      -1(P3)          ;PUT INDEX ON STACK
918 0525 C402           LDI     2               ;INCREMENT I.L. PC, SKIPPING
919 0527 02             CCL                     ; OVER TEST-FAIL ADDRESS
920 0528 F2FB           ADD     PCLOW(P2)
921 052A CAFB           ST      PCLOW(P2)
922 052C C400           LDI     0
923 052E F2FA           ADD     PCHIGH(P2)
924 0530 CAFA           ST      PCHIGH(P2)
925 0532 90AE           JMP     X10
926
927
928             ;************************************
929             ;*     IND -- EVALUATE A VARIABLE    *
930             ;************************************
931
```

```
 932 0584 C410    IND     LDI     H(AESTK)        ;SET P3 TO STACK
 933 0586 37              XPAH    P3
 934 0587 BAFD            ILD     LSTK(P2)
 935 0589 33              XPAL    P3
 936 058A C3FE            LD      -2(P3)          ;GET INDEX OFF TOP
 937 058C 01              XAE                     ;PUT INDEX IN E REG
 938 058D C280            LD      EREG(P2)        ;GET LOWER 8 BITS
 939 058F C3FF            ST      -2(P3)          ;SAVE ON STACK
 940 0591 02              CCL
 941 0592 40              LDE                     ;INCREMENT E REG
 942 0593 F401            ADI     1
 943 0595 01              XAE
 944 0596 C280            LD      EREG(P2)        ;GET UPPER 8 BITS
 945 0598 C3FF            ST      -1(P3)          ;SAVE ON STACK
 946 059A 9096    X11     JMP     X10
 947
 948
 949              ;*************************************
 950              ;*      RELATIONAL OPERATORS         *
 951              ;*************************************
 952
 953 059F C401    EQ      LDI     1               ;EACH RELATIONAL OPERATOR
 954 05A1 9012            JMP     CMP             ;  LOADS A NUMBER USED LATER
 955 05A3 C402    NEQ     LDI     2               ;  AS A CASE SELECTOR, AFTER
 956 05A5 900E            JMP     CMP             ;  THE TWO OPERANDS ARE COM-
 957 05A7 C403    LSS     LDI     3               ;  PARED.  BASED ON THE COM-
 958 05A9 9008            JMP     CMP             ;  PARISON, FLAGS ARE SET THAT
 959 05AB C404    LEQ     LDI     4               ;  ARE EQUIVALENT TO THOSE SET
 960 05AD 9004            JMP     CMP             ;  BY THE 'CMP' INSTRUCTION IN
 961 05AF C405    GTR     LDI     5               ;  THE PDP-11.  THESE PSEUDO-
 962 05B1 9002            JMP     CMP             ;  FLAGS ARE USED TO DETERMINE
 963 05B3 C406    GEQ     LDI     6               ;  WHETHER THE PARTICULAR
 964                                              ;  RELATION IS SATISFIED OR NO
 965 05B5 CAEB    CMP     ST      NUM(P2)
 966 05B7 C410            LDI     H(AESTK)        ;SET P3 -> ARITH STACK
 967 05B9 37              XPAH    P3
 968 05BA BAFD            DLD     LSTK(P2)
 969 05BC BAFD            DLD     LSTK(P2)
 970 05BE 33              XPAL    P3
 971 05BF 03              SCL
 972 05C0 C3FE            LD      -2(P3)          ;SUBTRACT THE TWO OPERANDS,
 973 05C2 FB00            CAD     (P3)            ;  STORING RESULT IN LO & HI
 974 05C4 CAEF            ST      LO(P2)
 975 05C6 C3FF            LD      -1(P3)
 976 05C8 FB01            CAD     1(P3)
 977 05CA CAEE            ST      HI(P2)
 978 05CC E3FF            XOR     -1(P3)          ;OVERFLOW OCCURS IF SIGNS OF
 979 05CE 01              XAE                     ;  RESULT AND 1ST OPERAND
 980 05CF C3FF            LD      -1(P3)          ;  DIFFER, AND SIGNS OF THE
 981 05D1 E301            XOR     1(P3)           ;  TWO OPERANDS DIFFER
 982 05D3 50              ANE                     ;BIT 7 EQUIVALENT TO V FLAG
 983 05D4 E2EE            XOR     HI(P2)          ;BIT 7 EQUIVALENT TO N XOR V
 984 05D6 CAEA            ST      TEMP(P2)        ;STORE IN TEMP
 985 05D8 C2EE            LD      HI(P2)          ;DETERMINE IF RESULT WAS ZERO
 986 05DA DAEF            OR      LO(P2)
 987 05DC 9802            JZ      SETZ            ;IF RESULT=0, SET Z FLAG
 988 05DE C480            LDI     080             ;  ELSE CLEAR Z FLAG
 989 05E0 E480    SETZ    XRI     080
 990 05E2 01              XAE                     ;BIT 7 OF EX = Z FLAG
 991
 992 05E3 BAEB            DLD     NUM(P2)         ;TEST FOR =
 993 05E5 9C05            JNZ     NE01
 994 05E7 40              LDE                     ;  EQUAL IF Z = 1
 995 05E8 902B            JMP     CMP1
 996 05EA 90B1    X12     JMP     X11
 997 05EC BAEB    NE01    DLD     NUM(P2)         ;TEST FOR <>
 998 05EE 9C05            JNZ     LSS1
 999 05F0 40              LDE                     ;  NOT EQUAL IF Z = 0
1000 05F1 E480            XRI     080
1001 05F3 9020            JMP     CMP1
1002 05F5 BAEB    LSS1    DLD     NUM(P2)         ;TEST FOR <
1003 05F7 9C04            JNZ     LE01
1004 05F9 C2EA            LD      TEMP(P2)        ;  LESS THAN IF (N XOR V)=1
```

```
1005 05A8 9018           JMP     CMP1
1006 05AA BAEB   LE01    DLD     NUM(P2)          ;TEST FOR <=
1007 05AC 9C05           JNZ     GTR1
1008 05AE 40             LDE                      ; LESS THAN OR EQUAL
1009 05AF DAEA           OR      TEMP(P2)         ;   IF (Z OR (N XOR V))=1
1010 05B1 900F           JMP     CMP1
1011 05B3 BAEB   GTR1.   DLD     NUM(P2)          ;TEST FOR >
1012 05B5 9C07           JNZ     GE01
1013 05B7 40             LDE                      ; GREATER THAN
1014 05B8 DAEA           OR      TEMP(P2)         ;   IF (Z OR (N XOR V))=0
1015 05BA E480           XRI     080
1016 05BC 9004           JMP     CMP1
1017 05BE C2EA   GE01.   LD      TEMP(P2)         ;GREATER THAN OR EQUAL
1018 05C0 E480           XRI     080              ; IF (N XOR V)=0
1019 05C2 9404   CMP1.   JP      FALSE            ;IS RELATION SATISFIED?
1020 05C4 C401           LDI     1                ;YES - PUSH 1 ON STACK
1021 05C6 9002           JMP     CMP2
1022 05C8 C400   FALSE.  LDI     0                ;NO - PUSH 0 ON STACK
1023 05CA CBFE   CMP2.   ST      -2(P3)
1024 05CC C400           LDI     0
1025 05CE CBFF           ST      -1(P3)
1026 05D0 C400           JS      P3,RTN           ;DO AN I.L. RETURN
1027 05D7 90BE           JMP     X12
1028
1029
1030                     ;************************************
1031                     ;*   IF STATEMENT TEST FOR ZERO     *
1032                     ;************************************
1033
1034 05D9 C2EF   CMPR    LD      LO(P2)           ;GET LOW & HI BYTES OF EXPR.
1035 05DB DAEE           OR      HI(P2)           ;TEST IF EXPRESSION IS ZERO
1036 05DD 9802           JZ      FAIL             ;YES - IT IS
1037 05DF 90B6           JMP     X12              ;NO - IT ISN'T SO CONTINUE
1038 05E1 C501   FAIL.   LD      @1(P1)           ;SKIP TO NEXT LINE IN PROGRAM
1039 05E3 E40D           XRI     0D               ; (I.E. TIL NEXT CR)
1040 05E5 9CFA           JNZ     FAIL
1041 05E7 C402           JS      P3,NXT           ;CALL NXT AND RETURN
1042 05EE 90A7   X12A:   JMP     X12
1043
1044
1045                     ;************************************
1046                     ;*        AND, OR, & NOT            *
1047                     ;************************************
1048
1049                     .LOCAL
1050 05F0 C401   ANDOP.  LDI     1                ;EACH OPERATION HAS ITS
1051 05F2 9006           JMP     $1               ; OWN CASE SELECTOR.
1052 05F4 C402   OROP.   LDI     2
1053 05F6 9002           JMP     $1
1054 05F8 C403   NOTOP.  LDI     3
1055 05FA CAEB   $1.     ST      NUM(P2)
1056 05FC C410           LDI     H(AESTK)         ;SET P3 -> ARITH. STACK
1057 05FE 37             XPAH    P3
1058 05FF BAFD           DLD     LSTK(P2)
1059 0601 BAFD           DLD     LSTK(P2)
1060 0603 33             XPAL    P3
1061 0604 BAEB           DLD     NUM(P2)          ;TEST FOR 'AND'
1062 0606 9C0E           JNZ     $OR
1063 0608 C301           LD      1(P3)            ;REPLACE TWO TOP ITEMS ON
1064 060A D3FF           AND     -1(P3)           ; STACK BY THEIR 'AND'
1065 060C CBFF           ST      -1(P3)
1066 060E C300           LD      0(P3)
1067 0610 D3FE           AND     -2(P3)
1068 0612 CBFE           ST      -2(P3)
1069 0614 90D8           JMP     X12A
1070 0616 BAEB   $OR:    DLD     NUM(P2)          ;TEST FOR 'OR'
1071 0618 9C0E           JNZ     $NOT
1072 061A C301           LD      1(P3)            ;REPLACE TWO TOP ITEMS ON
1073 061C DBFF           OR      -1(P3)           ; STACK BY THEIR 'OR'
1074 061E CBFF           ST      -1(P3)
1075 0620 C300           LD      0(P3)
1076 0622 DBFE           OR      -2(P3)
1077 0624 CBFE           ST      -2(P3)
```

```
1078 0626 90C6            JMP     X12A
1079 0628 C701    $NOT    LD      @1(P3)          ;'NOT' OPERATION
1080 062A E4FF            XRI     OFF
1081 062C CBFF            ST      -1(P3)          ;REPLACE TOP ITEM ON STACK
1082 062E C701            LD      @1(P3)          ; BY ITS ONE'S COMPLEMENT
1083 0630 E4FF            XRI     OFF
1084 0632 CBFF            ST      -1(P3)
1085 0634 33              XPAL    P3
1086 0635 CAFD            ST      LSTK(P2)        ;STACK POINTER FIXUP
1087 0637 90B5    X12B:   JMP     X12A
1088
1089
1090              ;************************************
1091              ;*     EXCHANGE CURSOR WITH RAM     *
1092              ;************************************
1093
1094 0639 C2F1    XCHGP1: LD      P1LOW(P2)       ;THIS ROUTINE IS HANDY WHEN
1095 063B 31              XPAL    P1              ; EXECUTING AN 'INPUT' STMT
1096 063C CAF1            ST      P1LOW(P2)       ; IT EXCHANGES THE CURRENT
1097 063E C2F0            LD      P1HIGH(P2)      ; TEXT CURSOR WITH ONE SAVED
1098 0640 35              XPAH    P1              ; IN RAM
1099 0641 CAF0            ST      P1HIGH(P2)
1100 0643 3F              XPPC    P3
1101
1102
1103              ;************************************
1104              ;*          CHECK RUN MODE          *
1105              ;************************************
1106
1107 0644 C2F4    CKMODE: LD      RUNMOD(P2)      ;THIS ROUTINE CAUSES AN ERROR
1108 0646 9801            JZ      CK1             ; IF CURRENTLY IN EDIT MODE
1109 0648 3F              XPPC    P3
1110 0649 C403    CK1:    LDI     3
1111 064B CAEB    E8:     ST      NUM(P2)         ;ERROR IF RUN MODE = 0
1112 064D C402            JS      P3,ERR2         ;MINOR KLUGE
1113
1114
1115              ;************************************
1116              ;*     GET HEXADECIMAL NUMBER       *
1117              ;************************************
1118
1119                       LOCAL
1120 0654 AAFD    HEX:    ILD     LSTK(P2)        ;POINT P3 AT ARITH STACK
1121 0656 AAFD            ILD     LSTK(P2)
1122 0658 33              XPAL    P3
1123 0659 C410            LDI     H(AESTK)
1124 065B 37              XPAH    P3
1125 065C C400            LDI     0               ;NUMBER INITIALLY ZERO
1126 065E CBFF            ST      -1(P3)          ;PUT IT ON STACK
1127 0660 CBFE            ST      -2(P3)
1128 0662 CAEB            ST      NUM(P2)         ;ZERO NUMBER OF DIGITS
1129 0664 C501    $SKIP:  LD      @1(P1)          ;SKIP ANY SPACES
1130 0666 E420            XRI     ' '
1131 0668 98FA            JZ      $SKIP
1132 066A C5FF            LD      @-1(P1)
1133 066C C100    $LOOP:  LD      (P1)            ;GET A CHARACTER
1134 066E 03              SCL
1135 066F FC3A            CAI     '9'+1           ;CHECK FOR A NUMERIC CHAR
1136 0671 9409            JP      $LETR
1137 0673 03              SCL
1138 0674 FCF6            CAI     '0'-'9'-1       ;IF NUMERIC, SHIFT NUMBER
1139 0676 9413            JP      $ENTER          ; AND ADD NEW HEX DIGIT
1140 0678 9032            JMP     $END
1141 067A 90BB    X12C:   JMP     X12B
1142 067C 03      $LETR:  SCL                     ;CHECK FOR HEX LETTER
1143 067D FC0D            CAI     'G'-'9'-1
1144 067F 942B            JP      $END
1145 0681 03              SCL
1146 0682 FCFA            CAI     'A'-'G'
1147 0684 9402            JP      $OK
1148 0686 9024            JMP     $END
1149 0688 02      $OK:    CCL                     ;ADD 10 TO GET TRUE VALUE
1150 0689 F40A            ADI     10              ; OF LETTER
1151 068B 01      $ENTER: XAE                     ;NEW DIGIT IN EX REG
```

```
1152 068C C404          LDI     4               ;SET SHIFT COUNTER
1153 068E CAEA          ST      TEMP(P2)
1154 0690 CAEB          ST      NUM(P2)         ;DIGIT COUNT IS NON-ZERO
1155 0692 C3FE  $SHIFT: LD      -2(P3)          ;SHIFT NUMBER LEFT BY 4
1156 0694 02            CCL
1157 0695 F3FE          ADD     -2(P3)
1158 0697 CBFE          ST      -2(P3)
1159 0699 C3FF          LD      -1(P3)
1160 069B F3FF          ADD     -1(P3)
1161 069D CBFF          ST      -1(P3)
1162 069F BAEA          DLD     TEMP(P2)
1163 06A1 9CEF          JNZ     $SHIFT
1164 06A3 C3FE          LD      -2(P3)          ;ADD NEW DIGIT
1165 06A5 58            ORE                     ; INTO NUMBER
1166 06A6 CBFE          ST      -2(P3)
1167 06A8 C501          LD      @1(P1)          ;ADVANCE THE CURSOR
1168 06AA 90C0          JMP     $LOOP           ;GET NEXT CHAR
1169 06AC C2EB  $END:   LD      NUM(P2)         ;CHECK IF THERE WERE
1170 06AE 9C87          JNZ     X12B            ; MORE THAN 0 CHARACTERS
1171 06B0 C405          LDI     5               ;ERROR IF THERE WERE NONE
1172 06B2 9097  E8B:    JMP     E8
1173
1174
1175                    ;*************************************
1176                    ;*       TEST FOR NUMBER IN TEXT       *
1177                    ;*************************************
1178
1179                    ; THIS ROUTINE TESTS FOR A NUMBER IN THE TEXT.  IF NO
1180                    ; NUMBER IS FOUND, I.L. CONTROL PASSES TO THE ADDRESS
1181                    ; INDICATED IN THE 'TSTN' INSTRUCTION.  OTHERWISE, THE
1182                    ; NUMBER IS SCANNED AND PUT ON THE ARITHMETIC STACK,
1183                    ; WITH I.L. CONTROL PASSING TO THE NEXT INSTRUCTION.
1184
1185                    .LOCAL
1186 06B4 C501  TSTNUM: LD      @1(P1)
1187 06B6 E420          XRI     ' '             ;SKIP OVER ANY SPACES
1188 06B8 98FA          JZ      TSTNUM
1189 06BA C5FF          LD      @-1(P1)         ;GET FIRST CHAR
1190 06BC 03            SCL                     ;TEST FOR DIGIT
1191 06BD FC3A          CAI     '9'+1
1192 06BF 9405          JP      $ABORT
1193 06C1 03            SCL
1194 06C2 FCF6          CAI     '0'-'9'-1
1195 06C4 9421          JP      $1
1196 06C6 C2FB  $ABORT: LD      PCLOW(P2)       ;GET TEST-FAIL ADDRESS
1197 06C8 33            XPAL    P3              ; FROM I.L. TABLE
1198 06C9 C2FA          LD      PCHIGH(P2)
1199 06CB 37            XPAH    P3
1200 06CC C300          LD      (P3)            ;PUT TEST-FAIL ADDRESS
1201 06CE CAFA          ST      PCHIGH(P2)      ; INTO I.L. PC
1202 06D0 C301          LD      1(P3)
1203 06D2 CAFB          ST      PCLOW(P2)
1204 06D4 90A4          JMP     X12C
1205 06D6 C402  $RET:   LDI     2               ;SKIP OVER ONE IL INSTRUCTION
1206 06D8 02            CCL                     ; IF NUMBER IS DONE
1207 06D9 F2FB          ADD     PCLOW(P2)
1208 06DB CAFB          ST      PCLOW(P2)
1209 06DD C400          LDI     0
1210 06DF F2FA          ADD     PCHIGH(P2)
1211 06E1 CAFA          ST      PCHIGH(P2)
1212 06E3 9095  X13:    JMP     X12C
1213 06E5 90CB  E8A:    JMP     E8B
1214 06E7 01    $1:     XAE                     ;SAVE DIGIT IN EX REG
1215 06E8 C410          LDI     H(AESTK)        ;POINT P3 AT AE STACK
1216 06EA 37            XPAH    P3
1217 06EB AAFD          ILD     LSTK(P2)
1218 06ED AAFD          ILD     LSTK(P2)
1219 06EF 33            XPAL    P3
1220 06F0 C400          LDI     0
1221 06F2 CBFF          ST      -1(P3)
1222 06F4 40            LDE
1223 06F5 CBFE          ST      -2(P3)
1224 06F7 C501  $LOOP:  LD      @1(P1)          ;GET NEXT CHAR
1225 06F9 C100          LD      (P1)
```

```
1226 06FB 03            SCL                     ;TEST IF IT IS DIGIT
1227 06FC FC3A          CAI     '9'+1
1228 06FE 94D6          JP      $RET            ;RETURN IF IT ISN'T
1229 0700 03            SCL
1230 0701 FCF6          CAI     '0'-'9'-1
1231 0703 9402          JP      $2
1232 0705 90CF          JMP     $RET
1233 0707 01    $2:     XAE                     ;SAVE DIGIT
1234 0708 C3FF          LD      -1(P3)          ;PUT RESULT IN SCRATCH SPACE
1235 070A CB01          ST      1(P3)
1236 070C C3FE          LD      -2(P3)
1237 070E CB00          ST      (P3)
1238 0710 C402          LDI     2
1239 0712 CAEA          ST      TEMP(P2)        ;MULTIPLY RESULT BY 10
1240 0714 02    $SHIFT: CCL                     ;FIRST MULTIPLY BY 4
1241 0715 C3FE          LD      -2(P3)
1242 0717 F3FE          ADD     -2(P3)
1243 0719 CBFE          ST      -2(P3)
1244 071B C3FF          LD      -1(P3)
1245 071D F3FF          ADD     -1(P3)
1246 071F CBFF          ST      -1(P3)
1247 0721 BAEA          DLD     TEMP(P2)
1248 0723 9CEF          JNZ     $SHIFT
1249 0725 02            CCL                     ;THEN ADD OLD RESULT,
1250 0726 C3FE          LD      -2(P3)          ;  SO WE HAVE RESULT * 5
1251 0728 F300          ADD     (P3)
1252 072A CBFE          ST      -2(P3)
1253 072C C3FF          LD      -1(P3)
1254 072E F301          ADD     1(P3)
1255 0730 CBFF          ST      -1(P3)
1256 0732 02            CCL                     ;THEN MULTIPLY BY TWO
1257 0733 C3FE          LD      -2(P3)
1258 0735 F3FE          ADD     -2(P3)
1259 0737 CBFE          ST      -2(P3)
1260 0739 C3FF          LD      -1(P3)
1261 073B F3FF          ADD     -1(P3)
1262 073D CBFF          ST      -1(P3)
1263 073F 02            CCL                     ;THEN ADD IN NEW DIGIT
1264 0740 40            LDE
1265 0741 F3FE          ADD     -2(P3)
1266 0743 CBFE          ST      -2(P3)
1267 0745 C400          LDI     0
1268 0747 F3FF          ADD     -1(P3)
1269 0749 CBFF          ST      -1(P3)
1270 074B 94AA          JP      $LOOP           ;REPEAT IF NO OVERFLOW
1271 074D C406          LDI     6
1272 074F 9094  E9:     JMP     E8A             ;ELSE REPORT ERROR
1273 0751 9090  X14:    JMP     X13
1274
1275
1276            ;************************************
1277            ;*    GET LINE FROM TELETYPE        *
1278            ;************************************
1279
1280              LOCAL
1281 0753       GETL:   LDPI    P1,LBUF         ;SET P1 TO LBUF
1282 0759 C400          LDI     0               ;CLEAR NO. OF CHAR
1283 075B CAE7          ST      CHRNUM(P2)
1284 075D               LDPI    P3,PUTC-1       ;POINT P3 AT PUTC ROUTINE
1285 0763 C2F4          LD      RUNMOD(P2)      ;PRINT '? ' IF RUNNING
1286 0765 9808          JZ      $0              ;  (I.E. DURING 'INPUT')
1287 0767 C43F          LDI     '?'
1288 0769 3F            XPPC    P3
1289 076A C420          LDI     ' '
1290 076C 3F            XPPC    P3
1291 076D 9003          JMP     $1
1292 076F C43E  $0:     LDI     '>'             ;OTHERWISE PRINT '>'
1293 0771 3F            XPPC    P3
1294 0772 C40F  $1:     JS      P3,GECO         ;GET CHARACTER
1295 0779 C4BD          LDI     L(PUTC)-1       ;POINT P3 AT PUTC AGAIN
1296 077B 33            XPAL    P3
1297 077C 40            LDE                     ;GET TYPED CHAR
1298 077D 98F3          JZ      $1              ;IGNORE NULLS
1299 077F E40A          XRI     0A              ;IGNORE LINE FEED
```

```
1300 0781 98EF            JZ      $1
1301 0783 40              LDE
1302 0784 E40D            XRI     0D                ;CHECK FOR CR
1303 0786 9850            JZ      $CR
1304 0788 40              LDE
1305 0789 E45F            XRI     'O'+010           ;CHECK FOR SHIFT/O
1306 078B 9841            JZ      $RUB
1307 078D 40              LDE                       ;CHECK FOR CTRL/H
1308 078E E408            XRI     8
1309 0790 9836            JZ      $XH
1310 0792 40              LDE
1311 0793 E415            XRI     015               ;CHECK FOR CTRL/U
1312 0795 980F            JZ      $XU
1313 0797 40              LDE
1314 0798 E403            XRI     3                 ;CHECK FOR CTRL/C
1315 079A 9C1A            JNZ     $ENTER
1316 079C C45E            LDI     '^'               ;ECHO CONTROL/C AS ^C
1317 079E 3F              XPPC    P3
1318 079F C443            LDI     'C'
1319 07A1 3F              XPPC    P3
1320 07A2 C40E            LDI     14                ;CAUSE A BREAK
1321 07A4 90A9            JMP     E9
1322 07A6 C45E  $XU:      LDI     '^'               ;ECHO CONTROL/U AS ^U
1323 07A8 3F              XPPC    P3
1324 07A9 C455            LDI     'U'
1325 07AB 3F              XPPC    P3
1326 07AC C40D            LDI     0D                ;PRINT CR/LF
1327 07AE 3F              XPPC    P3
1328 07AF C40A            LDI     0A
1329 07B1 3F              XPPC    P3
1330 07B2 909F  $2:       JMP     GETL              ;GO GET ANOTHER LINE
1331 07B4 909B  X15:      JMP     X14
1332 07B6 40    $ENTER:   LDE
1333 07B7 CD01            ST      @1(P1)            ;PUT CHAR IN LBUF
1334 07B9 AAE7            ILD     CHRNUM(P2)        ;INCREMENT CHRNUM
1335 07BB E448            XRI     72                ;IF=72, LINE FULL
1336 07BD 9CB3            JNZ     $1
1337 07BF C40D            LDI     0D
1338 07C1 01              XAE                       ;SAVE CARRIAGE RET
1339 07C2 40              LDE
1340 07C3 3F              XPPC    P3                ;PRINT IT
1341 07C4 9012            JMP     $CR               ;STORE IT IN LBUF
1342 07C6 9087  E10:      JMP     E9
1343 07C8 C420  $XH:      LDI     ' '               ;BLANK OUT THE CHARACTER
1344 07CA 3F              XPPC    P3
1345 07CB C408  <-        LDI     8                 ;PRINT ANOTHER BACKSPACE
1346 07CD 3F              XPPC    P3
1347 07CE C2E7  $RUB:     LD      CHRNUM(P2)
1348 07D0 98A0            JZ      $1
1349 07D2 BAE7            DLD     CHRNUM(P2)        ;ONE LESS CHAR
1350 07D4 C5FF            LD      @-1(P1)           ;BACKSPACE CURSOR
1351 07D6 909A            JMP     $1
1352 07D8 40    $CR:      LDE
1353 07D9 CD01            ST      @1(P1)            ;STORE CR IN LBUF
1354 07DB C40A            LDI     0A                ;PRINT LINE FEED
1355 07DD 3F              XPPC    P3
1356 07DE C410            LDI     H(LBUF)           ;SET P1 TO BEGIN-
1357 07E0 35              XPAH    P1                ; NING OF LBUF
1358 07E1 C4D6            LDI     L(LBUF)
1359 07E3 31              XPAL    P1
1360 07E4 90CE  X16:      JMP     X15
1361
1362
1363              ;************************************
1364              ;*      EVAL -- GET MEMORY CONTENTS     *
1365              ;************************************
1366
1367              ; THIS ROUTINE IMPLEMENTS THE '@' OPERATOR IN EXPRESSIONS
1368
1369 07E6 C410  EVAL:    LDI     H(AESTK)
1370 07E8 37              XPAH    P3
1371 07E9 C2FD            LD      LSTK(P2)
1372 07EB 33              XPAL    P3                ;P3 -> ARITH STACK
1373 07EC C3FF            LD      -1(P3)            ;GET ADDR OFF STACK,
```

```
1374 07EE 35            XPAH    P1              ; AND INTO P1,
1375 07EF 01            XAE                     ; SAVING OLD P1 IN EX & LO
1376 07F0 C3FE          LD      -2(P3)
1377 07F2 31            XPAL    P1
1378 07F3 CAEF          ST      LO(P2)
1379 07F5 C100          LD      0(P1)           ;GET MEMORY CONTENTS,
1380 07F7 CBFE          ST      -2(P3)          ; SHOVE ONTO STACK
1381 07F9 C400          LDI     0
1382 07FB CBFF          ST      -1(P3)          ;HIGH ORDER 8 BITS ZEROED
1383 07FD C2EF          LD      LO(P2)
1384 07FF 31            XPAL    P1              ;RESTORE ORIGINAL P1
1385 0800 40            LDE
1386 0801 35            XPAH    P1
1387 0802 90B0          JMP     X15
1388
1389
1390                ;*************************************
1391                ;*    MOVE -- STORE INTO MEMORY      *
1392                ;*************************************
1393
1394                ; THIS ROUTINE IMPLEMENTS THE STATEMENT:
1395                ;     '@' FACTOR '=' REL-EXP
1396
1397 0804 C410    MOVE:   LDI     H(AESTK)
1398 0806 37            XPAH    P3
1399 0807 C2FD          LD      LSTK(P2)
1400 0809 33            XPAL    P3              ;P3 -> ARITH STACK
1401 080A C7FE          LD      @-2(P3)         ;GET BYTE TO BE MOVED
1402 080C 01            XAE
1403 080D C7FF          LD      @-1(P3)         ;NOW GET ADDRESS INTO P3
1404 080F CAEA          ST      TEMP(P2)
1405 0811 C7FF          LD      @-1(P3)
1406 0813 33            XPAL    P3
1407 0814 CAFD          ST      LSTK(P2)        ;STACK PTR UPDATED NOW
1408 0816 C2EA          LD      TEMP(P2)
1409 0818 37            XPAH    P3
1410 0819 40            LDE
1411 081A CB00          ST      0(P3)           ;MOVE THE BYTE INTO MEMORY
1412 081C 90C6  X17:    JMP     X16
1413 081E 90A6  E11:    JMP     E10
1414
1415                ;*************************************
1416                ;*            TEXT EDITOR            *
1417                ;*************************************
1418
1419                ;INPUTS TO THIS ROUTINE: POINTER TO LINE BUFFER IN P1LOW &
1420                ; P1HIGH.  P1 POINTS TO THE INSERTION POINT IN THE TEXT.
1421                ; THE A.E. STACK HAS THE LINE NUMBER ON IT (STACK POINTER
1422                ; IS ALREADY POPPED).
1423
1424                ;EACH LINE IN THE NIBL TEXT IS STORED IN THE FOLLOWING
1425                ; FORMAT.  TWO BYTES CONTAINING THE LINE NUMBER (IN BINARY,
1426                ; HIGH ORDER BYTE FIRST), THEN ONE BYTE CONTAINING THE
1427                ; LENGTH OF THE LINE, AND FINALLY THE LINE ITSELF FOLLOWED
1428                ; BY A CARRIAGE RETURN.  THE LAST LINE IN THE TEXT IS
1429                ; FOLLOWED BY TWO CONSECUTIVE BYTES OF X'FF.
1430
1431                        .LOCAL
1432 0820 C410    INSRT:  LDI     H(AESTK)        ;POINT P3 AT AE STACK,
1433 0822 37            XPAH    P3              ; WHICH HAS THE LINE #
1434 0823 C2FD          LD      LSTK(P2)        ; ON IT
1435 0825 33            XPAL    P3
1436 0826 C301          LD      1(P3)           ;SAVE NEW LINE'S NUMBER
1437 0828 CAF7          ST      HILINE(P2)
1438 082A C300          LD      0(P3)
1439 082C CAF8          ST      LOLINE(P2)
1440 082E C2F1          LD      P1LOW(P2)       ;PUT POINTER TO LBUF INTO P3
1441 0830 33            XPAL    P3
1442 0831 C2F0          LD      P1HIGH(P2)
1443 0833 37            XPAH    P3
1444 0834 C404          LDI     4               ;INITIALLY LENGTH OF NEW LINE
1445 0836 CAE7          ST      CHRNUM(P2)      ; = 4.  ADD 1 TO LENGTH FOR
1446 0838 C701  $1:     LD      @1(P3)          ; EACH CHAR IN LINE UP TO, BU
```

```
1447 083A E40D          XRI     0D              ; NOT INCLUDING, CARR. RETURN
1448 083C 9804          JZ      $2
1449 083E AAE7          ILD     CHRNUM(P2)
1450 0840 90F6          JMP     $1
1451 0842 C2E7  $2:     LD      CHRNUM(P2)      ;IF LENGTH STILL 4, WE'LL DEL
1452 0844 E404          XRI     4               ; A LINE, SO SET LENGTH = 0
1453 0846 9C02          JNZ     $3
1454 0848 CAE7          ST      CHRNUM(P2)
1455 084A C2E7  $3:     LD      CHRNUM(P2)      ;PUT NEW LINE LENGTH IN EX
1456 084C 01            XAE
1457 084D C2F2          LD      LABLHI(P2)      ;IS NEW LINE REPLACING OLD?
1458 084F 9406          JP      $4              ;YES - DO REPLACE
1459 0851 D47F          ANI     07F             ;NO - WE'LL INSERT LINE HERE,
1460 0853 CAF2          ST      LABLHI(P2)      ; WHERE FNDLBL GOT US.
1461 0855 9018          JMP     $MOVE           ;BUT FIRST MAKE ROOM
1462 0857 C503  $4:     LD      @3(P1)          ;SKIP LINE # AND LENGTH
1463 0859 40            LDE                     ;EX, NOW HOLDING NEW LINE
1464 085A 02            CCL                     ; LENGTH, WILL SOON HOLD
1465 085B F4FC          ADI     -4              ; DISPLACEMENT OF LINES
1466 085D 01            XAE                     ; TO BE MOVED
1467 085E C501  $5:     LD      @1(P1)          ;SUBTRACT 1 FROM DISPLACEMENT
1468 0860 E40D          XRI     0D              ; FOR EACH CHAR IN LINE BEING
1469 0862 980B          JZ      $MOVE           ; REPLACED
1470 0864 40            LDE
1471 0865 02            CCL
1472 0866 F4FF          ADI     -1
1473 0868 01            XAE
1474 0869 90F3          JMP     $5
1475 086B 90AF  X19:    JMP     X17
1476 086D 90AF  E12:    JMP     E11
1477 086F 40    $MOVE:  LDE                     ;IF DISPLACEMENT AND LENGTH
1478 0870 DAE7          OR      CHRNUM(P2)      ; OF NEW LINE ARE 0, RETURN
1479 0872 98F7          JZ      X19
1480 0874 C47A          LDI     L(DOSTAK)       ;CLEAR SOME STACKS
1481 0876 CAFF          ST      DOPTR(P2)
1482 0878 C46A          LDI     L(SBRSTK)
1483 087A CAFC          ST      SBRPTR(P2)
1484 087C C48A          LDI     L(FORSTK)
1485 087E CAFE          ST      FORPTR(P2)
1486 0880 40            LDE
1487 0881 9860          JZ      $ADD            ;DON'T NEED TO MOVE LINES
1488 0883 9410          JP      $UP             ;SKIP IF DISPLACEMENT POSITIV
1489 0885 C100  $DOWN:  LD      0(P1)           ;NEGATIVE DISPLACEMENT:
1490 0887 C980          ST      EREG(P1)        ;DO;
1491 0889 C501          LD      @1(P1)          ;    M(P1+DISP) = M(P1);
1492 088B 94F8          JP      $DOWN           ;    P1 = P1+1;
1493 088D C100          LD      0(P1)           ;UNTIL M(P1)<0 & M(P1-1)<0;
1494 088F 94F4          JP      $DOWN
1495 0891 C980          ST      EREG(P1)        ;M(P1+DISP) = M(P1);
1496 0893 904E          JMP     $ADD
1497 0895 C1FE  $UP:    LD      -2(P1)          ;POSITIVE DISPLACEMENT:
1498 0897 CAEA          ST      TEMP(P2)        ;FLAG BEGINNING OF MOVE WITH
1499 0899 C4FF          LDI     -1              ; A -1 FOLLOWED BY 80, WHICH
1500 089B C9FE          ST      -2(P1)          ; CAN NEVER APPEAR IN A
1501 089D C450          LDI     80              ; NIBL TEXT
1502 089F C9FF          ST      -1(P1)
1503 08A1 C501  $UP1:   LD      @1(P1)          ;ADVANCE P1 TO END OF TEXT
1504 08A3 94FC          JP      $UP1
1505 08A5 C100          LD      0(P1)
1506 08A7 94F8          JP      $UP1
1507 08A9 35            XPAH    P1              ;SAVE P1 IN LO, HI
1508 08AA CAEE          ST      HI(P2)
1509 08AC 35            XPAH    P1
1510 08AD 31            XPAL    P1
1511 08AE CAEF          ST      LO(P2)
1512 08B0 31            XPAL    P1
1513 08B1 C2EF          LD      LO(P2)          ;ADD DISPLACEMENT TO
1514 08B3 02            CCL                     ; VALUE OF P1, TO CHECK
1515 08B4 70            ADE                     ; WHETHER WE'RE OUT OF
1516 08B5 C400          LDI     0               ; RAM FOR USER'S PROGRAM
1517 08B7 F2EE          ADD     HI(P2)
1518 08B9 E2EE          XOR     HI(P2)
1519 08BB D4F0          ANI     0F0
```

```
1520 08BD 9803           JZ      $UP2
1521 08BF C400           LDI     0                ;IF OUT OF RAM, CHANGE
1522 08C1 01             XAE                      ; DISPLACEMENT TO ZERO
1523 08C2 C4FF   $UP2:   LDI     -1
1524 08C4 C980   $UP3:   ST      EREG(P1)         ;MOVE TEXT UP UNTIL WE REACH
1525 08C6 C5FF           LD      @-1(P1)          ; THE FLAGS SET ABOVE
1526 08C8 94FA           JP      $UP3
1527 08CA C101           LD      1(P1)
1528 08CC E450           XRI     80
1529 08CE 9804           JZ      $UP4
1530 08D0 C100           LD      0(P1)
1531 08D2 90F0           JMP     $UP3
1532 08D4 C2EA   $UP4:   LD      TEMP(P2)         ;RESTORE THE FLAGGED LOCATION
1533 08D6 C900           ST      0(P1)            ; TO THEIR ORIGINAL VALUES
1534 08D8 C40D           LDI     0D
1535 08DA C901           ST      1(P1)
1536 08DC 40             LDE                      ;IF DISPLACEMENT = 0, WE'RE
1537 08DD 9C04           JNZ     $ADD             ; OUT OF RAM, SO REPORT ERROR
1538 08DF C402           LDI     2
1539 08E1 908A   E12A:   JMP     E12
1540 08E3 C2E7   $ADD:   LD      CHRNUM(P2)       ;INSERT NEW LINE
1541 08E5 9884   X19A:   JZ      X19              ; UNLESS LENGTH IS ZERO
1542 08E7 C2F1           LD      P1LOW(P2)        ;POINT P1 AT LINE BUFFER
1543 08E9 31             XPAL    P1
1544 08EA C2F0           LD      P1HIGH(P2)
1545 08EC 35             XPAH    P1
1546 08ED C2F3           LD      LABLLO(P2)       ;POINT P3 AT INSERTION PLACE
1547 08EF 33             XPAL    P3
1548 08F0 C2F2           LD      LABLHI(P2)
1549 08F2 37             XPAH    P3
1550 08F3 C2F7           LD      HILINE(P2)       ;PUT LINE NUMBER INTO TEXT
1551 08F5 CF01           ST      @1(P3)
1552 08F7 C2F8           LD      LOLINE(P2)
1553 08F9 CF01           ST      @1(P3)
1554 08FB C2E7           LD      CHRNUM(P2)       ;STORE LINE LENGTH IN TEXT
1555 08FD CF01           ST      @1(P3)
1556 08FF C501   $ADD1:  LD      @1(P1)           ;PUT REST OF CHARS
1557 0901 CF01           ST      @1(P3)           ; (INCLUDING CR) INTO TEXT
1558 0903 E40D           XRI     0D
1559 0905 9CF8           JNZ     $ADD1
1560 0907 90DC           JMP     X19A             ;RETURN
1561 0909 C400   X20:    JS      P3,EXECIL
1562 0910 90CF   E13:    JMP     E12A
1563
1564
1565                     ;************************************
1566                     ;*       POP ARITHMETIC STACK       *
1567                     ;************************************
1568
1569 0912 BAFD   POPAE:  DLD     LSTK(P2)         ;THIS ROUTINE POP THE A.E.
1570 0914 BAFD           DLD     LSTK(P2)         ; STACK, AND PUTS THE RESULT
1571 0916 33             XPAL    P3               ; INTO LO(P2) AND HI(P2)
1572 0917 C410           LDI     H(AESTK)
1573 0919 37             XPAH    P3
1574 091A C300           LD      (P3)
1575 091C CAEF           ST      LO(P2)
1576 091E C301           LD      1(P3)
1577 0920 CAEE           ST      HI(P2)
1578 0922 90E5           JMP     X20
1579
1580
1581                     ;************************************
1582                     ;*              UNTIL               *
1583                     ;************************************
1584
1585                      LOCAL
1586 0924 C2FF   UNTIL:  LD      DOPTR(P2)        ;CHECK FOR DO-STACK UNDERFLOW
1587 0926 01             XAE
1588 0927 40             LDE
1589 0928 E47A           XRI     L(DOSTAK)
1590 092A 9C04           JNZ     $1
1591 092C C40F           LDI     15
1592 092E 90E0           JMP     E13
1593 0930 C2EF   $1:     LD      LO(P2)           ;CHECK FOR EXPRESSION = 0
```

```
1594 0932 DAEE            OR      HI(P2)
1595 0934 9806            JZ      $REDO             ;IF ZERO, REPEAT DO-LOOP
1596 0936 BAFF            DLD     DOPTR(P2)         ;ELSE POP SAVE STACK
1597 0938 BAFF            DLD     DOPTR(P2)
1598 093A 90CD            JMP     X20               ;CONTINUE TO NEXT STMT
1599 093C 40      $REDO:  LDE                       ;POINT P3 AT DO-STACK
1600 093D 33              XPAL    P3
1601 093E C410            LDI     H(DOSTAK)
1602 0940 37              XPAH    P3
1603 0941 C3FF            LD      -1(P3)            ;LOAD P1 FROM DO STACK
1604 0943 35              XPAH    P1
1605 0944 C3FE            LD      -2(P3)
1606 0946 31              XPAL    P1                ;CURSOR NOW POINTS TO FIRST
1607 0947 90C0            JMP     X20               ; STATEMENT OF DO-LOOP
1608
1609
1610                ;*************************************
1611                ;*      STORE INTO STATUS REGISTER     *
1612                ;*************************************
1613
1614                ; THIS ROUTINE IMPLEMENTS THE STATEMENT:
1615                ;      'STAT' '=' REL-EXP
1616
1617 0949 C2EF  MOVESR: LD      LO(P2)            ;LOW BYTE GOES TO STATUS
1618 094B D4F7           ANI     0F7               ; BUT WITH IEN BIT CLEARED
1619 094D 07             CAS
1620 094E 90B9  X21:     JMP     X20
1621 0950 90BE  E14:     JMP     E13
1622
1623
1624                ;*************************************
1625                ;*          STAT FUNCTION            *
1626                ;*************************************
1627
1628 0952 C410  STATUS: LDI      H(AESTK)
1629 0954 37           XPAH     P3                ;POINT P3 AT AE STACK
1630 0955 AAFD         ILD      LSTK(P2)
1631 0957 AAFD         ILD      LSTK(P2)
1632 0959 33           XPAL     P3
1633 095A 06           CSA
1634 095B CBFE         ST       -2(P3)            ;STATUS REG IS LOW BYTE
1635 095D C400         LDI      0
1636 095F CBFF         ST       -1(P3)            ;ZERO IS HIGH BYTE
1637 0961 90EB         JMP      X21
1638
1639
1640                ;*************************************
1641                ;*    MACHINE LANGUAGE SUBROUTINE    *
1642                ;*************************************
1643
1644                ; THIS ROUTINE IMPLEMENTS THE 'LINK' STATEMENT
1645
1646 0963 C2EE  CALLML: LD       HI(P2)            ;GET HIGH BYTE OF ADDRESS
1647 0965 37           XPAH     P3
1648 0966 C2EF         LD       LO(P2)            ;GET LOW BYTE
1649 0968 33           XPAL     P3                ;P3 -> USER'S ROUTINE
1650 0969 C7FF         LD       @-1(P3)           ;CORRECT P3
1651 096B 3F           XPPC     P3                ;CALL ROUTINE (PRAY IT WORKS)
1652 096C              LDPI     P2,VARS           ;RESTORE RAM POINTER
1653 0972 90DA         JMP      X21               ;RETURN
1654
1655
1656                ;*************************************
1657                ;*         SAVE DO LOOP ADDRESS      *
1658                ;*************************************
1659
1660                ; THIS ROUTINE IMPLEMENTS THE 'DO' STATEMENT.
1661
1662                        LOCAL
1663 0974 C2FF  SAVEDO: LD       DOPTR(P2)         ;CHECK FOR STACK OVERFLOW
1664 0976 E48A           XRI      L(FORSTK)
1665 0978 9C04           JNZ      $1
1666 097A C40A           LDI      10
1667 097C 90D2  E15:     JMP      E14
```

```
1668 097E AAFF   $1:      ILD     DOPTR(P2)
1669 0980 AAFF            ILD     DOPTR(P2)
1670 0982 33              XPAL    P3
1671 0983 C410            LDI     H(DOSTAK)
1672 0985 37              XPAH    P3              ;P3 -> TOP OF DO STACK
1673 0986 35              XPAH    P1              ;SAVE CURSOR ON THE STACK
1674 0987 CBFF            ST      -1(P3)
1675 0989 35              XPAH    P1
1676 098A 31              XPAL    P1
1677 098B CBFE            ST      -2(P3)
1678 098D 31              XPAL    P1
1679 098E 90BE   X22:     JMP     X21
1680
1681
1682             ;*************************************
1683             ;*         TOP OF RAM FUNCTION       *
1684             ;*************************************
1685
1686                       .LOCAL
1687 0990 C2E9   TOP:     LD      TEMP2(P2)       ;SET P3 TO POINT TO
1688 0992 37              XPAH    P3              ; START OF NIBL TEXT
1689 0993 C2E8            LD      TEMP3(P2)
1690 0995 33              XPAL    P3
1691 0996 C300   $0:      LD      (P3)            ;HAVE WE HIT END OF TEXT?
1692 0998 9402            JP      $1              ;NO - SKIP TO NEXT LINE
1693 099A 9007            JMP     $2              ;YES - PUT CURSOR ON STACK
1694 099C C302   $1:      LD      2(P3)           ;GET LENGTH OF LINE
1695 099E 01              XAE
1696 099F C780            LD      @EREG(P3)       ;SKIP TO NEXT LINE
1697 09A1 90F3            JMP     $0              ;GO CHECK FOR EOF
1698 09A3 C702   $2:      LD      @2(P3)          ;P3 := P3 + 2
1699 09A5 AAFD            ILD     LSTK(P2)        ;SET P3 TO STACK, SAVING
1700 09A7 AAFD            ILD     LSTK(P2)        ; OLD P3 (WHICH CONTAINS TOP)
1701 09A9 33              XPAL    P3              ; ON IT SOMEHOW
1702 09AA 01              XAE
1703 09AB C410            LDI     H(AESTK)
1704 09AD 37              XPAH    P3
1705 09AE CBFF            ST      -1(P3)
1706 09B0 40              LDE
1707 09B1 CBFE            ST      -2(P3)
1708 09B3 90D9            JMP     X22
1709
1710
1711             ;*************************************
1712             ;*       SKIP TO NEXT NIBL LINE      *
1713             ;*************************************
1714
1715 09B5 C501   IGNORE:  LD      @1(P1)          ;SCAN TIL WE'RE PAST
1716 09B7 E40D            XRI     0D              ; CARRIAGE RETURN
1717 09B9 9CFA            JNZ     IGNORE
1718 09BB 3F              XPPC    P3              ;YES - RETURN
1719
1720
1721             ;*************************************
1722             ;*          MODULO FUNCTION          *
1723             ;*************************************
1724
1725 09BC C2FD   MODULO:  LD      LSTK(P2)        ;THIS ROUTINE MUST BE
1726 09BE 33              XPAL    P3              ; IMMEDIATELY AFTER A
1727 09BF C410            LDI     H(AESTK)        ; DIVIDE TO WORK CORRECTLY
1728 09C1 37              XPAH    P3
1729 09C2 C303            LD      3(P3)           ;GET LOW BYTE OF REMAINDER
1730 09C4 CBFE            ST      -2(P3)          ;PUT ON STACK
1731 09C6 C302            LD      2(P3)           ;GET HIGH BYTE OF REMAINDER
1732 09C8 CBFF            ST      -1(P3)          ;PUT ON STACK
1733 09CA 90C2   X23:     JMP     X22
1734 09CC 90AE   E16:     JMP     E15
1735
1736
1737             ;*************************************
1738             ;*          RANDOM FUNCTION          *
1739             ;*************************************
1740
1741                       LOCAL
```

```
1742 09CE C408   RANDOM: LDI     8              ;LOOP COUNTER FOR MULTIPLY
1743 09D0 CAEB           ST      NUM(P2)
1744 09D2 C2E5           LD      RNDX(P2)
1745 09D4 01             XAE
1746 09D5 C2E4           LD      RNDY(P2)
1747 09D7 CAE9           ST      TEMP2(P2)
1748 09D9 C2E5   $LOOP:  LD      RNDX(P2)       ;MULTIPLY THE SEEDS BY 9
1749 09DB 02             CCL
1750 09DC 70             ADE
1751 09DD 01             XAE
1752 09DE C2E4           LD      RNDY(P2)
1753 09E0 02             CCL
1754 09E1 F2E9           ADD     TEMP2(P2)
1755 09E3 CAE4           ST      RNDY(P2)
1756 09E5 BAEB           DLD     NUM(P2)
1757 09E7 9CF0           JNZ     $LOOP
1758 09E9 40             LDE                    ;ADD 7 TO SEEDS
1759 09EA 02             CCL
1760 09EB F407           ADI     7
1761 09ED 01             XAE
1762 09EE C2E4           LD      RNDY(P2)
1763 09F0 02             CCL
1764 09F1 F407           ADI     7
1765 09F3 1E             RR
1766 09F4 CAE4           ST      RNDY(P2)
1767 09F6 AAE6           ILD     RNDF(P2)       ;HAVE WE GONE THROUGH
1768 09F8 9803           JZ      $1             ; 256 GENERATIONS?
1769 09FA 40             LDE                    ;IF SO, SKIP GENERATING
1770 09FB CAE5           ST      RNDX(P2)       ; THE NEW RNDX
1771 09FD C2FD   $1:     LD      LSTK(P2)       ;START MESSING WITH THE STACK
1772 09FF 33             XPAL    P3
1773 0A00 C410           LDI     H(AESTK)
1774 0A02 37             XPAH    P3
1775 0A03 C401           LDI     1              ;FIRST PUT 1 ON STACK
1776 0A05 CB00           ST      (P3)
1777 0A07 C400           LDI     0
1778 0A09 CB01           ST      1(P3)
1779 0A0B C3FE           LD      -2(P3)         ;PUT EXPR2 ON STACK
1780 0A0D CB02           ST      2(P3)
1781 0A0F C3FF           LD      -1(P3)
1782 0A11 CB03           ST      3(P3)
1783 0A13 C3FC           LD      -4(P3)         ;PUT EXPR1 ON STACK
1784 0A15 CB04           ST      4(P3)
1785 0A17 C3FD           LD      -3(P3)
1786 0A19 CB05           ST      5(P3)
1787 0A1B C2E4           LD      RNDY(P2)       ;PUT RANDOM # ON STACK
1788 0A1D CBFE           ST      -2(P3)
1789 0A1F C2E5           LD      RNDX(P2)
1790 0A21 E4FF           XRI     0FF
1791 0A23 D47F           ANI     07F
1792 0A25 CBFF           ST      -1(P3)
1793 0A27 C706           LD      @6(P3)         ;ADD 6 TO STACK POINTER
1794 0A29 33             XPAL    P3
1795 0A2A CAFD           ST      LSTK(P2)
1796 0A2C 909C   X24     JMP     X23
1797 0A2E 909C   E16A    JMP     E16
1798
1799
1800                     ;*************************************
1801                     ;*     PUSH 1 ON ARITHMETIC STACK    *
1802                     ;*************************************
1803
1804 0A30 AAFD   LIT1    ILD     LSTK(P2)
1805 0A32 AAFD           ILD     LSTK(P2)
1806 0A34 33             XPAL    P3
1807 0A35 C410           LDI     H(AESTK)
1808 0A37 37             XPAH    P3
1809 0A38 C400           LDI     0
1810 0A3A CBFF           ST      -1(P3)
1811 0A3C C401           LDI     1
1812 0A3E CBFE           ST      -2(P3)
1813 0A40 90EA           JMP     X24
1814
1815
```

```
1816                    ;************************************
1817                    ;*      FOR-LOOP INITIALIZATION     *
1818                    ;************************************
1819
1820                            LOCAL
1821 0A42 C2FE    SAVFOR:  LD      FORPTR(P2)      ;CHECK FOR FOR STACK
1822 0A44 E4A6             XRI     L(PCSTAK)       ; OVERFLOW
1823 0A46 9C04             JNZ     $1
1824 0A48 C40A             LDI     10
1825 0A4A 90E2    E17:     JMP     E16A
1826 0A4C E4A6    $1:      XRI     L(PCSTAK)
1827 0A4E 31               XPAL    P1              ;POINT P1 AT FOR STACK
1828 0A4F CAF1             ST      P1LOW(P2)       ; SAVING OLD P1
1829 0A51 C410             LDI     H(FORSTK)
1830 0A53 35               XPAH    P1
1831 0A54 CAF0             ST      P1HIGH(P2)
1832 0A56 C2FD             LD      LSTK(P2)        ;POINT P3 AT AE STACK
1833 0A58 33               XPAL    P3
1834 0A59 C410             LDI     H(AESTK)
1835 0A5B 37               XPAH    P3
1836 0A5C C3F9             LD      -7(P3)          ;GET VARIABLE INDEX
1837 0A5E CD01             ST      @1(P1)          ;SAVE ON FOR-STACK
1838 0A60 C3FC             LD      -4(P3)          ;GET L(LIMIT)
1839 0A62 CD01             ST      @1(P1)          ;SAVE
1840 0A64 C3FD             LD      -3(P3)          ;GET H(LIMIT)
1841 0A66 CD01             ST      @1(P1)          ;SAVE
1842 0A68 C3FE             LD      -2(P3)          ;GET L(STEP)
1843 0A6A CD01             ST      @1(P1)          ;SAVE
1844 0A6C C3FF             LD      -1(P3)          ;GET H(STEP)
1845 0A6E CD01             ST      @1(P1)          ;SAVE
1846 0A70 C2F1             LD      P1LOW(P2)       ;GET L(P1)
1847 0A72 CD01             ST      @1(P1)          ;SAVE
1848 0A74 C2F0             LD      P1HIGH(P2)      ;GET H(P1)
1849 0A76 CD01             ST      @1(P1)          ;SAVE
1850 0A78 35               XPAH    P1              ;RESTORE OLD P1
1851 0A79 C2F1             LD      P1LOW(P2)
1852 0A7B 31               XPAL    P1
1853 0A7C CAFE             ST      FORPTR(P2)      ;UPDATE FOR STACK PTR
1854 0A7E C7FC             LD      @-4(P3)
1855 0A80 33               XPAL    P3
1856 0A81 CAFD             ST      LSTK(P2)        ;UPDATE AE STACK PTR
1857 0A83 90A7    X25:     JMP     X24
1858
1859
1860                    ;************************************
1861                    ;*    FIRST PART OF 'NEXT VAR'      *
1862                    ;************************************
1863
1864                            .LOCAL
1865 0A85 C2FE    NEXTV:   LD      FORPTR(P2)      ;POINT P1 AT FOR STACK,
1866 0A87 E48A             XRI     L(FORSTK)       ; CHECKING FOR UNDERFLOW
1867 0A89 9C04             JNZ     $1
1868 0A8B C40B             LDI     11              ;REPORT ERROR
1869 0A8D 90BB             JMP     E17
1870 0A8F E48A    $1:      XRI     L(FORSTK)
1871 0A91 31               XPAL    P1
1872 0A92 CAF1             ST      P1LOW(P2)       ;SAVE OLD P1
1873 0A94 C410             LDI     H(FORSTK)
1874 0A96 35               XPAH    P1
1875 0A97 CAF0             ST      P1HIGH(P2)
1876 0A99 C2FD             LD      LSTK(P2)        ;POINT P3 AT AE STACK
1877 0A9B 33               XPAL    P3
1878 0A9C C410             LDI     H(AESTK)
1879 0A9E 37               XPAH    P3
1880 0A9F C7FF             LD      @-1(P3)         ;GET VARIABLE INDEX
1881 0AA1 E1F9             XOR     -7(P1)          ;COMPARE WITH INDEX
1882 0AA3 9804             JZ      $10             ; ON FOR STACK: ERROR
1883 0AA5 C40C             LDI     12              ; IF NOT EQUAL
1884 0AA7 90A1    E18:     JMP     E17
1885 0AA9 E1F9    $10:     XOR     -7(P1)          ;RESTORE INDEX
1886 0AAB 01               XAE                     ;SAVE IN EREG
1887 0AAC C280             LD      EREG(P2)        ;GET L(VARIABLE)
1888 0AAE 02               CCL
1889 0AAF F1FC             ADD     -4(P1)          ;ADD L(STEP)
```

```
1890 0AB1 CA80          ST      EREG(P2)        ;STORE IN VARIABLE
1891 0AB3 CB00          ST      (P3)            ; AND ON STACK
1892 0AB5 C601          LD      @1(P2)          ;INCREMENT RAM PTR
1893 0AB7 C280          LD      EREG(P2)        ;GET H(VARIABLE)
1894 0AB9 F1FD          ADD     -3(P1)          ;ADD H(STEP)
1895 0ABB CA80          ST      EREG(P2)        ;STORE IN VARIABLE
1896 0ABD CB01          ST      1(P3)           ; AND ON STACK
1897 0ABF C6FF          LD      @-1(P2)         ;RESTORE RAM POINTER
1898 0AC1 C1FA          LD      -6(P1)          ;GET L(LIMIT)
1899 0AC3 CB02          ST      2(P3)           ;PUT ON STACK
1900 0AC5 C1FB          LD      -5(P1)          ;GET H(LIMIT)
1901 0AC7 CB03          ST      3(P3)           ;PUT ON STACK
1902 0AC9 C1FD          LD      -3(P1)          ;GET H(STEP)
1903 0ACB 9410          JP      $2              ;IF NEGATIVE, INVERT
1904 0ACD C404          LDI     4               ; ITEMS ON A.E. STACK
1905 0ACF CAEB          ST      NUM(P2)         ;NUM = LOOP COUNTER
1906 0AD1 C701  $LOOP:  LD      @1(P3)          ;GET BYTE FROM STACK
1907 0AD3 E4FF          XRI     OFF             ;INVERT IT
1908 0AD5 CBFF          ST      -1(P3)          ;PUT BACK ON STACK
1909 0AD7 BAEB          DLD     NUM(P2)         ;DO UNTIL NUM = 0
1910 0AD9 9CF6          JNZ     $LOOP
1911 0ADB 9002          JMP     $3
1912 0ADD C704  $2:     LD      @4(P3)          ;UPDATE AE STACK POINTER
1913 0ADF 33    $3:     XPAL    P3
1914 0AE0 CAFD          ST      LSTK(P2)
1915 0AE2 C2F1          LD      P1LOW(P2)       ;RESTORE OLD P1
1916 0AE4 31            XPAL    P1
1917 0AE5 C2F0          LD      P1HIGH(P2)
1918 0AE7 35            XPAH    P1
1919 0AE8 9099  X26:    JMP     X25
1920
1921
1922            ;************************************
1923            ;*      SECOND PART OF 'NEXT VAR'       *
1924            ;************************************
1925
1926 0AEA C2EF  NEXTV1: LD      LO(P2)          ;IS FOR-LOOP OVER WITH?
1927 0AEC 9808          JZ      $REDO           ;NO - REPEAT LOOP
1928 0AEE C2FE          LD      FORPTR(P2)      ;YES - POP FOR-STACK
1929 0AF0 02            CCL
1930 0AF1 F4F9          ADI     -7
1931 0AF3 CAFE          ST      FORPTR(P2)
1932 0AF5 3F            XPPC    P3              ;RETURN TO I.L. INTERPRETER
1933 0AF6 C2FE  $REDO:  LD      FORPTR(P2)      ;POINT P3 AT FOR STACK
1934 0AF8 33            XPAL    P3
1935 0AF9 C410          LDI     H(FORSTK)
1936 0AFB 37            XPAH    P3
1937 0AFC C3FF          LD      -1(P3)          ;GET OLD P1 OFF STACK
1938 0AFE 35            XPAH    P1
1939 0AFF C3FE          LD      -2(P3)
1940 0B01 31            XPAL    P1
1941 0B02 90E4          JMP     X26
1942 0B04 90A1  E19:    JMP     E18
1943
1944
1945            ;************************************
1946            ;*      PRINT MEMORY AS STRING          *
1947            ;************************************
1948
1949            ; THIS ROUTINE IMPLEMENTS THE STATEMENT:
1950            ;       'PRINT' '$' FACTOR
1951
1952                    .LOCAL
1953 0B06 C2EE  PSTRNG: LD      HI(P2)          ;POINT P1 AT STRING TO PRINT
1954 0B08 35            XPAH    P1
1955 0B09 C2EF          LD      LO(P2)
1956 0B0B 31            XPAL    P1
1957 0B0C               LDPI    P3,PUTC-1       ;POINT P3 AT PUTC ROUTINE
1958 0B12 C501  $1:     LD      @1(P1)          ;GET A CHARACTER
1959 0B14 E40D          XRI     0D              ;IS IT A CARRIAGE RETURN?
1960 0B16 98D0          JZ      X26             ;YES - WE'RE DONE
1961 0B18 E40D          XRI     0D              ;NO - PRINT THE CHARACTER
1962 0B1A 3F            XPPC    P3
```

```
1963 0B1B 06            CSA                     ;MAKE SURE NO ONE IS
1964 0B1C D420          ANI     020             ; TYPING ON THE TTY
1965 0B1E 9CF2          JNZ     $1              ; BEFORE REPEATING LOOP
1966 0B20 90C6          JMP     X26
1967
1968
1969            ;************************************
1970            ;*         INPUT A STRING           *
1971            ;************************************
1972
1973            ; THIS ROUTINE IMPLEMENTS THE STATEMENT:
1974            ;       'INPUT' '$' FACTOR
1975
1976 0B22 C2EE  ISTRNG: LD      HI(P2)          ;GET ADDRESS TO STORE THE
1977 0B24 37            XPAH    P3              ; STRING, PUT IT INTO P3
1978 0B25 C2EF          LD      LO(P2)
1979 0B27 33            XPAL    P3
1980 0B28 C501  $2:     LD      @1(P1)          ;GET A BYTE FROM LINE BUFFER
1981 0B2A CF01          ST      @1(P3)          ;PUT IT IN SPECIFIED LOCATION
1982 0B2C E40D          XRI     0D              ;DO UNTIL CHAR = CARR. RETURN
1983 0B2E 9CF8          JNZ     $2
1984 0B30 90B6  X27:    JMP     X26
1985
1986
1987            ;************************************
1988            ;*   STRING CONSTANT ASSIGNMENT     *
1989            ;************************************
1990
1991            ; THIS ROUTINE IMPLEMENTS THE STATEMENT:
1992            ;       '$' FACTOR '=' STRING
1993
1994                    .LOCAL
1995 0B32 C2EF  PUTSTR: LD      LO(P2)          ;GET ADDRESS TO STORE STRING,
1996 0B34 33            XPAL    P3              ; PUT IT INTO P3
1997 0B35 C2EE          LD      HI(P2)
1998 0B37 37            XPAH    P3
1999 0B38 C501  $LOOP:  LD      @1(P1)          ;GET A BYTE FROM STRING
2000 0B3A E422          XRI     '"'             ;CHECK FOR END OF STRING
2001 0B3C 980E          JZ      $END
2002 0B3E E42F          XRI     '"' ! 0D        ;MAKE SURE THERE'S NO CR
2003 0B40 9C04          JNZ     $1
2004 0B42 C407          LDI     7
2005 0B44 90BE          JMP     E19             ;ERROR IF CARRIAGE RETURN
2006 0B46 E40D  $1:     XRI     0D              ;RESTORE CHARACTER
2007 0B48 CF01          ST      @1(P3)          ;PUT IN SPECIFIED LOCATION
2008 0B4A 90EC          JMP     $LOOP           ;GET NEXT CHARACTER
2009 0B4C C40D  $END:   LDI     0D              ;APPEND CARRIAGE RETURN
2010 0B4E CB00          ST      (P3)            ; TO STRING
2011 0B50 90DE          JMP     X27
2012
2013
2014            ;************************************
2015            ;*          MOVE STRING             *
2016            ;************************************
2017
2018            ; THIS ROUTINE IMPLEMENTS THE STATEMENT:
2019            ;       '$' FACTOR '=' '$' FACTOR
2020
2021                    .LOCAL
2022 0B52 C2FD  MOVSTR: LD      LSTK(P2)        ;POINT P3 AT A.E. STACK
2023 0B54 33            XPAL    P3
2024 0B55 C410          LDI     H(AESTK)
2025 0B57 37            XPAH    P3
2026 0B58 C7FF          LD      @-1(P3)         ;GET ADDRESS OF SOURCE STRING
2027 0B5A 35            XPAH    P1              ; INTO P1
2028 0B5B C7FF          LD      @-1(P3)
2029 0B5D 31            XPAL    P1
2030 0B5E C7FF          LD      @-1(P3)         ;GET ADDRESS OF DESTINATION
2031 0B60 01            XAE                     ; STRING INTO P3
2032 0B61 C7FF          LD      @-1(P3)
2033 0B63 33            XPAL    P3
2034 0B64 CAFD          ST      LSTK(P2)        ;UPDATE STACK POINTER
2035 0B66 40            LDE
```

```
0B67 37          $LOOP:  XPAH    P3
0B68 C501                LD      @1(P1)          ;GET A SOURCE CHARACTER
0B6A CF01                ST      @1(P3)          ;SEND IT TO DESTINATION
0B6C E40D                XRI     0D              ;REPEAT UNTIL CARRIAGE RET.
0B6E 98C0                JZ      X27
0B70 06                  CSA                     ; OR KEYBOARD INTERRUPT
0B71 D420                ANI     020
0B73 9CF3                JNZ     $LOOP
0B75 90B9                JMP     X27


                 ;************************************
                 ;*      PUT PAGE NUMBER ON STACK    *
                 ;************************************

0B77 AAFD        PUTPGE: ILD      LSTK(P2)
0B79 AAFD                ILD     LSTK(P2)
0B7B 33                  XPAL    P3
0B7C C410                LDI     H(AESTK)
0B7E 37                  XPAH    P3
0B7F C2F6                LD      PAGE(P2)
0B81 CBFE                ST      -2(P3)
0B83 C400                LDI     0
0B85 CBFF                ST      -1(P3)
0B87 90A7                JMP     X27


                 ;************************************
                 ;*         ASSIGN NEW PAGE          *
                 ;************************************

                         .LOCAL
0B89 C2EF        NUPAGE: LD       LO(P2)          ;GET PAGE # FROM STACK,
0B8B D407                ANI     7               ; GET THE LOW 3 BITS
0B8D 9C02                JNZ     $0              ;PAGE 0 BECOMES PAGE 1
0B8F C401                LDI     1
0B91 CAF6        $0:     ST      PAGE(P2)
0B93 3F                  XPPC    P3              ;RETURN


                 ;************************************
                 ;*          FIND START OF PAGE      *
                 ;************************************

                 ; THIS ROUTINE COMPUTES THE START OF THE CURRENT TEXT PAGE,
                 ; STORING THE ADDRESS IN TEMP2(P2) [THE HIGH BYTE], AND
                 ; TEMP3(P2) [THE LOW BYTE].

0B94 C2F6        FNDPGE: LD       PAGE(P2)
0B96 E401                XRI     1               ;SPECIAL CASE IS PAGE 1, BUT
0B98 9C09                JNZ     $1              ; OTHERS ARE CONVENTIONAL
0B9A C411                LDI     H(PGM)          ;PAGE 1 STARTS AT 'PGM'
0B9C CAE9                ST      TEMP2(P2)
0B9E C420                LDI     L(PGM)
0BA0 CAE8                ST      TEMP3(P2)
0BA2 3F                  XPPC    P3              ;RETURN
0BA3 E401        $1:     XRI     1               ;RESTORE PAGE #
0BA5 01                  XAE                     ;SAVE IT
0BA6 C404                LDI     4               ;LOOP COUNTER = 4
0BA8 CAEB                ST      NUM(P2)
0BAA 40          $LOOP:  LDE                     ;MULTIPLY PAGE# BY 16
0BAB 02                  CCL
0BAC 70                  ADE
0BAD 01                  XAE
0BAE BAEB                DLD     NUM(P2)
0BB0 9CF8                JNZ     $LOOP
0BB2 40                  LDE
0BB3 CAE9                ST      TEMP2(P2)       ;TEMP2 HAS HIGH BYTE
0BB5 C402                LDI     2               ; OF ADDRESS NOW
0BB7 CAE8                ST      TEMP3(P2)       ;LOW BYTE IS ALWAYS 2
0BB9 3F                  XPPC    P3
```

```
2109                ;*************************************
2110                ;*       MOVE CURSOR TO NEW PAGE     *
2111                ;*************************************
2112
2113 0BBA C2E9    CHPAGE: LD      TEMP2(P2)       ;PUT START OF PAGE
2114 0BBC 35            XPAH    P1              ; INTO P1.  THIS ROUTINE
2115 0BBD C2E8          LD      TEMP3(P2)       ; MUST BE CALLED RIGHT
2116 0BBF 31            XPAL    P1              ; AFTER 'FNDPGE'
2117 0BC0 3F            XPPC    P3              ;RETURN
2118
2119
2120                ;*************************************
2121                ;*       DETERMINE CURRENT PAGE      *
2122                ;*************************************
2123
2124 0BC1 35      DETPGE: XPAH    P1              ;CURRENT PAGE IS HIGH
2125 0BC2 01            XAE                     ; PART OF CURSOR DIVIDED
2126 0BC3 40            LDE                     ; BY 16
2127 0BC4 35            XPAH    P1
2128 0BC5 40            LDE
2129 0BC6 1C            SR
2130 0BC7 1C            SR
2131 0BC8 1C            SR
2132 0BC9 1C            SR
2133 0BCA CAF6          ST      PAGE(P2)
2134 0BCC 3F            XPPC    P3              ;RETURN
2135
2136
2137                ;*************************************
2138                ;*          CLEAR CURRENT PAGE       *
2139                ;*************************************
2140
2141 0BCD C2E9    NEWPGM: LD      TEMP2(P2)       ;POINT P1 AT CURRENT PAGE
2142 0BCF 35            XPAH    P1
2143 0BD0 C2E8          LD      TEMP3(P2)
2144 0BD2 31            XPAL    P1
2145 0BD3 C40D          LDI     0D              ;PUT DUMMY END-OF-LINE
2146 0BD5 C9FF          ST      -1(P1)          ; JUST BEFORE TEXT
2147 0BD7 C4FF          LDI     -1              ;PUT -1 AT START OF TEXT
2148 0BD9 C900          ST      (P1)
2149 0BDB C901          ST      1(P1)
2150 0BDD 3F            XPPC    P3              ;RETURN
2151
2152
2153                ;*************************************
2154                ;*       FIND LINE NUMBER IN TEXT    *
2155                ;*************************************
2156
2157                ;INPUTS:  THE START OF THE CURRENT PAGE IN TEMP2 AND TEMP3,
2158                ;         THE LINE NUMBER TO LOOK FOR IN LO AND HI.
2159                ;OUPUTS:  THE ADDRESS OF THE FIRST LINE IN THE NIBL TEXT
2160                ;         WHOSE LINE NUMBER IS GREATER THAN OR EQUAL TO THE
2161                ;         NUMBER IN HI AND LO, RETURNED IN ADRLO AND ADRHI
2162
2163                        LOCAL
2164 0BDE C2E9    FNDLBL  LD      TEMP2(P2)       ;POINT P1 AT START OF TEXT
2165 0BE0 35            XPAH    P1
2166 0BE1 C2E8          LD      TEMP3(P2)
2167 0BE3 31            XPAL    P1
2168 0BE4 C100    $1    LD      (P1)            ;HAVE WE HIT END OF TEXT?
2169 0BE6 E4FF          XRI     0FF
2170 0BE8 9412          JP      $2              ;YES - STOP LOOKING
2171 0BEA 03            SCL                     ;NO - COMPARE LINE NUMBERS
2172 0BEB C101          LD      1(P1)           ; BY SUBTRACTING
2173 0BED FAEF          CAD     LO(P2)
2174 0BEF C100          LD      0(P1)
2175 0BF1 FAEE          CAD     HI(P2)          ;IS TEXT LINE # >= LINE #?
2176 0BF3 9407          JP      $2              ;YES - STOP LOOKING
2177 0BF5 C102          LD      2(P1)           ;NO - TRY NEXT LINE IN TEXT
2178 0BF7 01            XAE
2179 0BF8 C580          LD      @EREG(P1)       ; SKIP LENGTH OF LINE
2180 0BFA 90E8          JMP     $1
2181 0BFC 31      $2    XPAL    P1              ;SAVE ADDRESS OF FOUND LINE
```

```
2182 0BFD CAF3            ST      LABLLO(P2)      , IN LABLHI AND LABLLO
2183 0BFF 31              XPAL    P1
2184 0C00 35              XPAH    P1
2185 0C01 CAF2            ST      LABLHI(P2)
2186 0C03 35              XPAH    P1
2187 0C04 C2EF            LD      LO(P2)          ;WAS THERE AN EXACT MATCH?
2188 0C06 E101            XOR     1(P1)
2189 0C08 9C07            JNZ     $3
2190 0C0A C2EE            LD      HI(P2)
2191 0C0C E100            XOR     0(P1)
2192 0C0E 9C01            JNZ     $3              ;NO - FLAG THE ADDRESS
2193 0C10 3F              XPPC    P3              ;YES - RETURN NORMALLY
2194 0C11 C2F2    $3      LD      LABLHI(P2)      ;SET SIGN BIT OF HIGH PART
2195 0C13 DC80            ORI     080             ; OF ADDRESS TO INDICATE
2196 0C15 CAF2            ST      LABLHI(P2)      ; INEXACT MATCH OF LINE #'S
2197 0C17 3F              XPPC    P3
2198
2199                      PAGE    ' I L MACROS'
2200
2201              ;**********************************
2202              ;*          I L MACROS            *
2203              ;**********************************
2204
2205                      LOCAL
2206
2207      2000    $TSTBIT =       TSTBIT*256
2208      8000    $CALBIT =       CALBIT*256
2209      4000    $JMPBIT =       JMPBIT*256
2210
2211                      MACRO   TST,FAIL,A,B
2212                      DBYTE   $TSTBIT!FAIL
2213                      IF      #=2
2214                      BYTE     A'!080
2215                      ELSE
2216                      ASCII   'A'
2217                      BYTE     B'!080
2218                      ENDIF
2219                      ENDM
2220
2221                      MACRO   TSTCR,FAIL
2222                      DBYTE   $TSTBIT!FAIL
2223                      BYTE    0D'!080
2224                      ENDM
2225
2226                      MACRO   TSTV,FAIL
2227                      ADDR    TSTVAR
2228                      DBYTE   FAIL
2229                      ENDM
2230
2231                      MACRO   TSTN,FAIL
2232                      ADDR    TSTNUM
2233                      DBYTE   FAIL
2234                      ENDM
2235
2236                      MACRO   JUMP,ADR
2237                      DBYTE   $JMPBIT!ADR
2238                      ENDM
2239
2240                      MACRO   CALL,ADR
2241                      DBYTE   $CALBIT!ADR
2242                      ENDM
2243
2244                      MACRO   DO
2245                      MLOC    I
2246                      SET     I,1
2247                      DO      #
2248                      ADDR    #I
2249                      SET     I,I+1
2250                      ENDDO
2251                      ENDM
2252                      PAGE    ' I L TABLE'
2253
2254              ;**********************************
```

```
2255                    ;*              I. L. TABLE             *
2256                    ;*************************************
2257
2258 0C18  START:  DO      NLINE
2259 0C1A  PROMPT: DO      GETL
2260 0C1C          TSTCR   PRMPT1
2261 0C1F          JUMP    PROMPT
2262 0C21  PRMPT1: TSTN    LIST
2263 0C25          DO      FNDPGE,XCHGP1,POPAE,FNDLBL,INSRT
2264 0C2F          JUMP    PROMPT
2265
2266 0C31  LIST:   TST     RUN,'LIS','T'
2267 0C37          DO      FNDPGE
2268 0C39          TSTN    LIST1
2269 0C3D          DO      POPAE,FNDLBL
2270 0C41          JUMP    LIST2
2271 0C43  LIST1   DO      CHPAGE
2272 0C45  LIST2   DO      LST
2273 0C47  LIST3:  CALL    PRNUM
2274 0C49          DO      LST3
2275 0C4B          JUMP    START
2276
2277 0C4D  RUN:    TST     CLR,'RU','N'
2278 0C52          DO      DONE
2279 0C54  BEGIN   DO      FNDPGE,CHPAGE,STRT,NXT
2280
2281 0C5C  CLR:    TST     NEW,'CLEA','R'
2282 0C63          DO      DONE,CLEAR,NXT
2283
2284 0C69  NEW     TST     STMT,'NE','W'
2285 0C6E          TSTN    DFAULT
2286 0C72          JUMP    NEW1
2287 0C74  DFAULT: DO      LIT1
2288 0C76  NEW1:   DO      DONE,POPAE,NUPAGE,FNDPGE,NEWPGM,NXT
2289
2290 0C82  STMT    TST     LET,'LE','T'
2291 0C87  LET:    TSTV    AT
2292 0C8B          TST     SYNTAX,'='
2293 0C8E          CALL    RELEXP
2294 0C90          DO      STORE,DONE,NXT
2295 0C96  AT:     TST     IF,'@'
2296 0C99          CALL    FACTOR
2297 0C9B          TST     SYNTAX,'='
2298 0C9E          CALL    RELEXP
2299 0CA0          DO      MOVE,DONE,NXT
2300
2301 0CA6  IF:     TST     UNT,'I','F'
2302 0CAA          CALL    RELEXP
2303 0CAC          TST     IF1,'THE','N'
2304 0CB2  IF1:    DO      POPAE,CMPR
2305 0CB6          JUMP    STMT
2306
2307 0CB8  UNT:    TST     DO,'UNTI','L'
2308 0CBF          DO      CKMODE
2309 0CC1          CALL    RELEXP
2310 0CC3          DO      DONE,POPAE,UNTIL,DETPGE,NXT
2311
2312 0CCD  DO      TST     GOTO,'D','O'
2313 0CD1          DO      CKMODE,DONE,SAVEDO,NXT
2314
2315 0CD9  GOTO    TST     RETURN,'G','O'
2316 0CDD          TST     GOSUB,'T','O'
2317 0CE1          CALL    RELEXP
2318 0CE3          DO      DONE
2319 0CE5          JUMP    GO1
2320 0CE7  GOSUB   TST     SYNTAX,'SU','B'
2321 0CEC          CALL    RELEXP
2322 0CEE          DO      DONE,SAV
2323 0CF2  GO1:    DO      FNDPGE,POPAE,FNDLBL,XFER,NXT
2324
2325 0CFC  RETURN  TST     NEXT,'RETUR','N'
2326 0D04          DO      DONE,RSTR,DETPGE,NXT
2327
2328 0D0C  NEXT:   TST     FOR,'NEX','T'
```

```
2329 0D12                DO       CKMODE
2330 0D14                TSTV     SYNTAX
2331 0D18                DO       DONE,NEXTV
2332 0D1C                CALL     GTROP
2333 0D1E                DO       POPAE,NEXTV1,DETPGE,NXT
2334
2335 0D26          FOR:  TST      STAT,'FO','R'
2336 0D2B                DO       CKMODE
2337 0D2D                TSTV     SYNTAX
2338 0D31                TST      SYNTAX,'='
2339 0D34                CALL     RELEXP
2340 0D36                TST      SYNTAX,'T','O'
2341 0D3A                CALL     RELEXP
2342 0D3C                TST      FOR1,'STE','P'
2343 0D42                CALL     RELEXP
2344 0D44                JUMP     FOR2
2345 0D46          FOR1  DO       LIT1
2346 0D48          FOR2  DO       DONE,SAVFOR,STORE,NXT
2347
2348 0D50          STAT  TST      PGE,'STA','T'
2349 0D56                TST      SYNTAX,'='
2350 0D59                CALL     RELEXP
2351 0D5B                DO       POPAE,MOVESR
2352 0D5F                DO       DONE,NXT
2353
2354 0D63          PGE   TST      DOLLAR,'PAG','E'
2355 0D69                TST      SYNTAX,'='
2356 0D6C                CALL     RELEXP
2357 0D6E                DO       DONE,POPAE,NUPAGE,FNDPGE,CHPAGE,NXT
2358
2359 0D7A          DOLLAR: TST    PRINT,'$'
2360 0D7D                CALL     FACTOR
2361 0D7F                TST      SYNTAX,'='
2362 0D82                TST      DOLR1,'"'
2363 0D85                DO       POPAE,PUTSTR
2364 0D89                JUMP     DOLR2
2365 0D8B          DOLR1 TST      SYNTAX,'$'
2366 0D8E                CALL     FACTOR
2367 0D90                DO       XCHGP1,MOVSTR,XCHGP1
2368 0D96          DOLR2 DO       DONE,NXT
2369
2370 0D9A          PRINT: TST     INPUT,'P','R'
2371 0D9E                TST      PR1,'IN','T'
2372 0DA3          PR1:  TST      PR2,'"'
2373 0DA6                DO       PRS
2374 0DA8                JUMP     COMMA
2375 0DAA          PR2:  TST      PR3,'$'
2376 0DAD                CALL     FACTOR
2377 0DAF                DO       XCHGP1,POPAE,PSTRNG,XCHGP1
2378 0DB7                JUMP     COMMA
2379 0DB9          PR3:  CALL     RELEXP
2380 0DBB                CALL     PRNUM
2381 0DBD          COMMA: TST     PR4,','
2382 0DC0                JUMP     PR1
2383 0DC2          PR4:  TST      PR5,','
2384 0DC5                JUMP     PR6
2385 0DC7          PR5   DO       NLINE
2386 0DC9          PR6   DO       DONE,NXT
2387
2388 0DCD          INPUT  TST     END,'INPU','T'
2389 0DD4                 DO      CKMODE
2390 0DD6                 TSTV    IN2
2391 0DDA                 DO      XCHGP1,GETL
2392 0DDE          IN1:   CALL    RELEXP
2393 0DE0                 DO      STORE,XCHGP1
2394 0DE4                 TST     IN3,','
2395 0DE7                 TSTV    SYNTAX
2396 0DEB                 DO      XCHGP1
2397 0DED                 TST     SYNTAX,','
2398 0DF0                 JUMP    IN1
2399 0DF2          IN2:   TST     SYNTAX,'$'
2400 0DF5                 CALL    FACTOR
2401 0DF7                 DO      XCHGP1,GETL,POPAE,ISTRNG,XCHGP1
2402 0E01          IN3:   DO      DONE,NXT
```

```
2403
2404 0E05          END:    TST     ML,'EN','D'
2405 0E0A                  DO      DONE,BREAK
2406
2407 0E0E          ML:     TST     REM,'LIN','K'
2408 0E14                  CALL    RELEXP
2409 0E16                  DO      DONE,XCHGP1,POPAE,CALLML,XCHGP1,NXT
2410
2411 0E22          REM:    TST     SYNTAX,'RE','M'
2412 0E27                  DO      IGNORE,NXT
2413
2414 0E2B          SYNTAX: DO      ERR
2415 0E2D          ERRNUM: CALL    PRNUM
2416 0E2F                  DO      FIN
2417
2418                ; NOTE. EACH RELATIONAL OPERATOR (EQ, LEQ, ETC.) DOES AN
2419                ; AUTOMATIC 'RTN' (THIS SAVES VALUABE BYTES!)
2420
2421 0E31          RELEXP: CALL    EXPR
2422 0E33                  TST     REL1,'='
2423 0E36                  CALL    EXPR
2424 0E38                  DO      EQ
2425 0E3A          REL1:   TST     REL4,'<'
2426 0E3D                  TST     REL2,'='
2427 0E40                  CALL    EXPR
2428 0E42                  DO      LEQ
2429 0E44          REL2:   TST     REL3,'>'
2430 0E47                  CALL    EXPR
2431 0E49                  DO      NEQ
2432 0E4B          REL3:   CALL    EXPR
2433 0E4D                  DO      LSS
2434 0E4F          REL4:   TST     RETEXP,'>'
2435 0E52                  TST     REL5,'='
2436 0E55                  CALL    EXPR
2437 0E57                  DO      GEQ
2438 0E59          REL5:   CALL    EXPR
2439 0E5B          GTROP:  DO      GTR
2440
2441 0E5D          EXPR:   TST     EX1,'-'
2442 0E60                  CALL    TERM
2443 0E62                  DO      NEG
2444 0E64                  JUMP    EX3
2445 0E66          EX1:    TST     EX2,'+'
2446 0E69          EX2:    CALL    TERM
2447 0E6B          EX3:    TST     EX4,'+'
2448 0E6E                  CALL    TERM
2449 0E70                  DO      ADD
2450 0E72                  JUMP    EX3
2451 0E74          EX4:    TST     EX5,'-'
2452 0E77                  CALL    TERM
2453 0E79                  DO      SUB
2454 0E7B                  JUMP    EX3
2455 0E7D          EX5:    TST     RETEXP,'O','R'
2456 0E81                  CALL    TERM
2457 0E83                  DO      OROP
2458 0E85                  JUMP    EX3
2459 0E87          RETEXP: DO      RTN
2460
2461 0E89          TERM:   CALL    FACTOR
2462 0E8B          T1:     TST     T2,'*'
2463 0E8E                  CALL    FACTOR
2464 0E90                  DO      MUL
2465 0E92                  JUMP    T1
2466 0E94          T2:     TST     T3,'/'
2467 0E97                  CALL    FACTOR
2468 0E99                  DO      DIV
2469 0E9B                  JUMP    T1
2470 0E9D          T3:     TST     RETEXP,'AN','D'
2471 0EA2                  CALL    FACTOR
2472 0EA4                  DO      ANDOP
2473 0EA6                  JUMP    T1
2474
2475 0EA8          FACTOR: TSTV    F1
```

```
2476 0EAC                DO       IND,RTN
2477 0EB0          F1:   TSTN     F2
2478 0EB4                DO       RTN
2479 0EB6          F2:   TST      F3,'#'
2480 0EB9                DO       HEX,RTN
2481 0EBD          F3:   TST      F4,'('
2482 0EC0                CALL     RELEXP
2483 0EC2                TST      SYNTAX,')'
2484 0EC5                DO       RTN
2485 0EC7          F4:   TST      F5,'@'
2486 0ECA                CALL     FACTOR
2487 0ECC                DO       EVAL,RTN
2488 0ED0          F5:   TST      F6,'NO','T'
2489 0ED5                CALL     FACTOR
2490 0ED7                DO       NOTOP,RTN
2491 0ED8          F6:   TST      F7,'STA','T'
2492 0EE1                DO       STATUS,RTN
2493 0EE5          F7:   TST      F8,'TO','P'
2494 0EEA                DO       FNDPGE,TOP,RTN
2495 0EF0          F8:   TST      F9,'MO','D'
2496 0EF5                CALL     DOUBLE
2497 0EF7                DO       DIV,MODULO,RTN
2498 0EFD          F9:   TST      F10,'RN','D'
2499 0F02                CALL     DOUBLE
2500 0F04                DO       RANDOM,SUB,ADD,DIV,MODULO,ADD,RTN
2501 0F12          F10:  TST      SYNTAX,'PAG','E'
2502 0F18                DO       PUTPGE,RTN
2503
2504 0F1C          DOUBLE: TST    SYNTAX,'('
2505 0F1F                CALL     RELEXP
2506 0F21                TST      SYNTAX,','
2507 0F24                CALL     RELEXP
2508 0F26                TST      SYNTAX,')'
2509 0F29                DO       RTN
2510
2511 0F2B          PRNUM:  DO     XCHGP1,PRN
2512 0F2F          PRNUM1: DO     DIV,PRN1,XCHGP1,RTN
2513                       PAGE   'ERROR MESSAGES'
2514
2515               ;*************************************
2516               ;*              ERROR MESSAGES       *
2517               ;*************************************
2518
2519                       .MACRO  MESSAGE,A,B
2520                       .ASCII  'A'
2521                       .BYTE   'B'!080
2522                       .ENDM
2523
2524 0F37          MESGS:  MESSAGE ' ERRO','R'      ; 1
2525 0F3D                  MESSAGE 'ARE','A'        ; 2
2526 0F41                  MESSAGE 'STM','T'        ; 3
2527 0F45                  MESSAGE 'CHA','R'        ; 4
2528 0F49                  MESSAGE 'SNT','X'        ; 5
2529 0F4D                  MESSAGE 'VAL','U'        ; 6
2530 0F51                  MESSAGE 'END','"'        ; 7
2531 0F55                  MESSAGE 'NOG','O'        ; 8
2532 0F59                  MESSAGE 'RTR','N'        ; 9
2533 0F5D                  MESSAGE 'NES','T'        ; 10
2534 0F61                  MESSAGE 'NEX','T'        ; 11
2535 0F65                  MESSAGE 'FO','R'         ; 12
2536 0F68                  MESSAGE 'DIV','O'        ; 13
2537 0F6C                  MESSAGE 'BR','K'         ; 14
2538 0F6F                  MESSAGE 'UNT','L'        ; 15
2539                       PAGE   ' TELETYPE ROUTINES'
2540
2541               ;*************************************
2542               ;*     GET CHARACTER AND ECHO IT     *
2543               ;*************************************
2544               ;
2545                       .LOCAL
2546 0F73 C408    GECO:   LDI    8                 ;SET COUNT = 8
2547 0F75 CAEB            ST     NUM(P2)
2548 0F77 06              CSA                      ;SET READER RELAY
```

```
2549 0F78 DC02           ORI     2
2550 0F7A 07             CAS
2551 0F7B 06     $1      CSA                     ;WAIT FOR START BIT
2552 0F7C D420           ANI     020
2553 0F7E 9CFB           JNZ     $1              ;NOT FOUND
2554 0F80 C457           LDI     87              ;DELAY 1/2 BIT TIME
2555 0F82 8F04           DLY     4
2556 0F84 06             CSA                     ;IS START BIT STILL THERE?
2557 0F85 D420           ANI     020
2558 0F87 9CF2           JNZ     $1              ;NO
2559 0F89 06             CSA                     ;SEND START BIT
2560 0F8A D4FD           ANI     %2              ;RESET READER RELAY
2561 0F8C DC01           ORI     1
2562 0F8E 07             CAS
2563 0F8F C485   $2:     LDI     133             ;DELAY 1 BIT TIME
2564 0F91 8F08           DLY     8
2565 0F93 06             CSA                     ;GET BIT (SENSEB)
2566 0F94 D420           ANI     020
2567 0F96 9804           JZ      $3
2568 0F98 C401           LDI     1
2569 0F9A 9004           JMP     $4
2570 0F9C C400   $3:     LDI     0
2571 0F9E 9C00           JNZ     $4
2572 0FA0 CAEA   $4:     ST      TEMP(P2)        ;SAVE BIT VALUE (0 OR 1)
2573 0FA2 1F             RRL                     ;ROTATE INTO LINK
2574 0FA3 01             XAE
2575 0FA4 1D             SRL                     ;SHIFT INTO CHARACTER
2576 0FA5 01             XAE                     ;RETURN CHAR TO E
2577 0FA6 06             CSA                     ;ECHO BIT TO OUTPUT
2578 0FA7 DC01           ORI     1
2579 0FA9 E2EA           XOR     TEMP(P2)
2580 0FAB 07             CAS
2581 0FAC BAEB           DLD     NUM(P2)         ;DECREMENT BIT COUNT
2582 0FAE 9CDF           JNZ     $2              ;LOOP UNTIL 0
2583 0FB0 06             CSA                     ;SET STOP BIT
2584 0FB1 D4FE           ANI     0FE
2585 0FB3 07             CAS
2586 0FB4 8F08           DLY     8               ;DELAY APPROX. 2 BIT TIMES
2587 0FB6 40             LDE                     ;AC HAS INPUT CHARACTER
2588 0FB7 D47F           ANI     07F
2589 0FB9 01             XAE
2590 0FBA 40             LDE
2591 0FBB 3F             XPPC    P3              ;RETURN
2592 0FBC 90B5           JMP     GECO
2593
2594                ;************************************
2595                ;*      PRINT CHARACTER AT TTY      *
2596                ;************************************
2597
2598 0FBE 01     PUTC:   XAE
2599 0FBF C4FF           LDI     255
2600 0FC1 8F17           DLY     23
2601 0FC3 06             CSA                     ;SET OUTPUT BIT TO LOGIC 0
2602 0FC4 DC01           ORI     1               ; FOR START BIT. (NOTE INVERS
2603 0FC6 07             CAS
2604 0FC7 C409           LDI     9               ;INITIALIZE BIT COUNT
2605 0FC9 CAE8           ST      TEMP3(P2)
2606 0FCB C48A   PUTC1:  LDI     138             ;DELAY 1 BIT TIME
2607 0FCD 8F08           DLY     8
2608 0FCF BAE8           DLD     TEMP3(P2)       ;DECREMENT BIT COUNT.
2609 0FD1 9810           JZ      PUTC2
2610 0FD3 40             LDE                     ;PREPARE NEXT BIT
2611 0FD4 D401           ANI     1
2612 0FD6 CAE9           ST      TEMP2(P2)
2613 0FD8 01             XAE                     ;SHIFT DATA RIGHT 1 BIT
2614 0FD9 1C             SR
2615 0FDA 01             XAE
2616 0FDB 06             CSA                     ;SET UP OUTPUT BIT
2617 0FDC DC01           ORI     1
2618 0FDE E2E9           XOR     TEMP2(P2)
2619 0FE0 07             CAS                     ;PUT BIT TO TTY
2620 0FE1 90E8           JMP     PUTC1
2621 0FE3 06     PUTC2:  CSA                     ;SET STOP BIT
2622 0FE4 D4FE           ANI     0FE
```

```
2623 0FE6 07              CAS
2624 0FE7 3F              XPPC    P3                  ;RETURN
2625 0FE8 90D4            JMP     PUTC
2626      0000             END    0
```

TELETYPE ROUTINES

ADD	0335	AESTK	1050	ANDOP	05F0	AT	0C96
BEGIN	0C54	BREAK	0288	CALBIT	0080	CALLML	0963
CHEAT	007C	CHEAT1	009B	CHPAGE	0BBA	CHRNUM	FFE7
CK1	0649	CKMODE	0644	CLEAR	0051	CLEAR1	0056
CLR	0C5C	CMP	0562	CMP1	05C2	CMP2	05CA
CMPR	05D9	COMMA	0DBD	DETPGE	0BC1	DFAULT	0C74
DIV	0410	DO	0CCD	DOLLAR	0D7A	DOLR1	0D8B
DOLR2	0D96	DONE	0135	DONE1	0143	DONE2	0144
DOPTR	FFFF	DOSTAK	107A	DOUBLE	0F1C	E0	0150
E0A	010D	E1	0195	E10	07C6	E11	081E
E12	086D	E12A	08E1	E13	0910	E14	0950
E15	097C	E16	09CC	E16A	0A2E	E17	0A4A
E18	0AA7	E19	0B04	E2	01CC	E3A	028A
E4	02DF	E5	0300	E6	0378	E6A	03D2
E8	064B	E8A	06E5	E8B	06B2	E9	074F
END	0E05	EO	054C	EREG	FF80	ERR	0223
ERR1	0225	ERR2	0227	ERRNUM	0E2D	EVAL	07E6
EX1	0E66	EX2	0E69	EX3	0E6B	EX4	0E74
EX5	0E7D	EXECIL	0076	EXPR	0E5D	F1	0EB0
F10	0F12	F2	0EB6	F3	0EBD	F4	0EC7
F5	0ED0	F6	0EDB	F7	0EE5	F8	0EF0
F9	0EFD	FACTOR	0EA8	FAIL	03E1	FAILHI	FFEC
FAILLO	FFED	FALSE	0508	FIN	02B2	FNULBL	0BDE
FNDPGE	0B94	FOR	0D26	FOR1	0D46	FOR2	0D48
FORPTR	FFFE	FORSTK	108A	GECO	0F73	GEQ	0560
GE01	05BE	GETL	0753	GO1	0CF2	GOSUB	0CE7
GOTO	0CD9	GTR	055C	GTR1	05B3	GTROP	0E5B
HEX	0654	HI	FFEB	HILINE	FFF7	IF	0CA6
IF1	0CB2	IGNORE	09B5	ILC1	00AA	ILCALL	00A0
IN1	0DDE	IN2	0DF2	IN3	0E01	IND	0534
INPUT	0DCD	INSRT	0820	ISTRNG	0B22	JMPBIT	0040
LABLHI	FFF2	LABLLO	FFF3	LDOP	10D6	LEQ	0558
LE01	05AA	LET	0C87	LIST	0C31	LIST1	0C43
LIST2	0C45	LIST3	0C47	LISTNG	FFF5	LIT1	0A30
LO	FFEF	LOLINE	FFF8	LSS	0554	LSS1	05A2
LST	02E1	LST2	02FF	LST3	030E	LST4	0314
LST5	0324	LSTK	FFFD	MESGS	0F37	ML	0E0E
MODULO	09BC	MOVE	0804	MOVESR	0949	MOVSTR	0B52
MUL	037A	NEG	0363	NEO	0550	NE01	0599
NEW	0C69	NEW1	0C76	NEWPGM	0BCD	NEXT	0D0C
NEXTV	0A85	NEXTV1	0AEA	NLINE	0215	NOJUMP	009D
NOTOP	05F8	NUM	FFEB	NUPAGE	0B89	NXT	028C
NXT1	02A9	OPOP	05F4	P1	0001	P1HIGH	FFF0
P1LOW	FFF1	P2	0002	P3	0003	PAGE	FFF6
PCHIGH	FFFA	PCLOW	FFFB	POSTAK	10A6	POSTK	FFF9
PGE	0D63	PGM	1120	POPAE	0912	PR1	0DA3
PR2	0DAA	PR3	0DB9	PR4	0DC2	PR5	0DC7
PR6	0DC9	PRINT	0D9A	PRMPT1	0C21	PRN	0197
PRN1	01CE	PRNUM	0F2B	PRNUM1	0F2F	PROMPT	0C1A
PRS	017E	PRS1	0193	PSTRNG	0B06	PUTC	0FBE
PUTC1	0FCB	PUTC2	0FE3	PUTPGE	0B77	PUTSTR	0B32
RANDOM	09CE	REL1	0E3A	REL2	0E44	REL3	0E4B
REL4	0E4F	REL5	0E59	RELEXP	0E31	REM	0E22
RETEXP	0E87	RETURN	0CFC	RNDF	FFE6	RNDX	FFE5
RNDY	FFE4	RSTR	0148	RSTR1	0152	RSTR2	0167
RTN	00FB	RUN	0C4D	RUNMOD	FFF4	SAV	010F
SAV1	012B	SAV2	0131	SAVEDO	0974	SAVFOR	0A42
SBRPTR	FFFC	SBRSTK	106A	SETZ	058D	START	0C18
STAT	0D50	STATUS	0972	STMT	0C82	STORE	04C9
STRT	02C8	SUB	034C	SYNTAX	0E2B	T1	0E8B
T2	0E94	T3	0E9D	TEMP	FFEA	TEMP2	FFE9
TEMP3	FFE8	TERM	0E89	TOP	0990	TST	00C5
TSTBIT	0020	TSTNUM	06B4	TSTVAR	04E9	UNT	0CB8
UNTIL	0924	VARS	101C	X0	00EC	X1	0165
X10	04E2	X11	054A	X12	0597	X12A	05EE
X12B	0637	X12C	067A	X13	06E3	X14	0751

```
X15     07B4      X16     07F4      X17     081C      X19     086B
X19A    08E5      X20     0909      X21     094E      X22     098E
X23     09CA      X24     0A2C      X25     0A83      X26     0AE8
X27     0B30      X4      01CA      X5      0221      X5A     0286
X6      02DD      X6A     030A      X7      034A      X8      0376
X9      03EF      X9A     0439      X9B     04A4      XCHGF1  0639
XFER    0171      XFER1   0179      ZZ0001  101C      ZZ0002  1120
ZZ0003  0FBD      ZZ0004  0FBD      ZZ0005  0FBD      ZZ0006  0FBD
ZZ0007  0F37      ZZ0008  0F37      ZZ0009  0FBD      ZZ000A  10D6
ZZ000B  0FBD      ZZ000C  101C      ZZ000D  0FBD      ZZ000E  0002
ZZ000F  0002      ZZ0010  0006      ZZ0011  0002      ZZ0012  0003
ZZ0013  0002      ZZ0014  0002      ZZ0015  0002      ZZ0016  0002
ZZ0017  0005      ZZ0018  0004      ZZ0019  0002      ZZ001A  0007
ZZ001B  0004      ZZ001C  0004      ZZ001D  0003      ZZ001E  0002
ZZ001F  0006      ZZ0020  0005      ZZ0021  0002      ZZ0022  0003
ZZ0023  0006      ZZ0024  0005      ZZ0025  0002      ZZ0026  0003
ZZ0027  0005      ZZ0028  0002      ZZ0029  0002      ZZ002A  0005
ZZ002B  0003      ZZ002C  0003      ZZ002D  0007      ZZ002E  0003
ZZ002F  0004      ZZ0030  0003      ZZ0031  0002      ZZ0032  0005
ZZ0033  0002      ZZ0034  0003      ZZ0035  0002      ZZ0036  0003
ZZ0037  0003      ZZ0038  0002      ZZ0039  0006      ZZ003A  0003
ZZ003B  0003      ZZ003C  0007      ZZ003D  0003      ZZ003E  0002
ZZ003F  0002      ZZ0040  0002      ZZ0041  0002      ZZ0042  0002
ZZ0043  0002      ZZ0044  0002      ZZ0045  0002      ZZ0046  0002
ZZ0047  0002      ZZ0048  0002      ZZ0049  0002      ZZ004A  0002
ZZ004B  0002      ZZ004C  0002      ZZ004D  0002      ZZ004E  0003
ZZ004F  0002      ZZ0050  0003      ZZ0051  0002      ZZ0052  0003
ZZ0053  0003      ZZ0054  0003      ZZ0055  0004      ZZ0056  0004
ZZ0057  0008      ZZ0058  0003      ZZ0059  0002      ZZ005A  0003
ZZ005B  0005      $0      002A      $0      0420      $0      076F
$0      0996      $0      0B91      $1      0043      $1      01C6
$1      023D      $1      0397      $1      0443      $1      05FA
$1      06E7      $1      0772      $1      0838      $1      0930
$1      097E      $1      099C      $1      09FD      $1      0A4C
$1      0A8F      $1      0B12      $1      0B46      $1      0BA3
$1      0BE4      $1      0F7B      $10     0AA9      $2      01FF
$2      025A      $2      03A8      $2      0454      $2      0707
$2      07B2 *    $2      0842      $2      09A3      $2      0ADD
$2      0B28      $2      0BFC      $2      0F8F      $3      0261
$3      03D4      $3      04AC      $3      084A      $3      0ADF
$3      0C11      $3      0F9C      $4      03F1      $4      0857
$4      0FA0      $5      08CE      $ABOR   06C6      $ADD    08E3
$ADD1   08FF      $CALB   8000      $CR     07D8      $DOWN   0885
$END    04C3      $END    06AC      $END    0B4C      $ENT1   04A6
$ENTE   068B      $ENTE   07B6      $EXIT   0402      $FAIL   04FB
$JMPB   4000      $LETR   067C      $LOOP   002C      $LOOP   00DB
$LOOP   0203      $LOOP   0241      $LOOP   03B6      $LOOP   0460
$LOOP   066C      $LOOP   06F7      $LOOP   09D9      $LOOP   0AD1
$LOOP   0B38      $LOOP   0B68      $LOOP   0BAA      $MAYB   050D
$MOVE   086F      $MSG    0247      $NE0    00EE      $NOT    0628
$OK     051A      $OK     0688      $OR     0616      $POS    043B
$PRNT   01EF      $REDO   093C      $REDO   0AF6      $RET    06D6
$RUB    07CE      $SCAN   00C7      $SHIF   0692      $SHIF   0714
$SKIP   0664      $TSTB   2000      $UP     0895      $UP1    08A1
$UP2    08C2      $UP3    08C4      $UP4    08D4      $XH     07C8
$XU     07A6

NO ERROR LINES
SOURCE CHECKSUM = 33FE
INPUT FILE     1 NIBL1127 SRC
```

Chapter 4

The Great Experiment Floppy ROM™ #1 Robert Uiterwyk's 6800 4K BASIC

By William Turner and William Blomgren

The one thing that characterizes the microcomputer industry, is the number of innovations that appear on a daily basis. During the infancy of the industry, *Interface Age* in May of 1977 kept pace with the introduction of portable software; in the form of a plastic record called a Floppy ROM™.

This article is a chronicle of the first Floppy ROM™ with Robert Uiterwyk's 4K BASIC for the 6800. What made this first record significant was that it was the first time major systems software was published in machine readable form, that just about anyone could use. More importantly it opened up an era of cheap software distribution.

WHY "PLATTER BASIC" ON A FLOPPY-ROM™?

This is really a two-part question. First, why BASIC at all? The need for high level software is fairly self-evident, and BASIC will be explained in a second article later in this issue. For now, let's just settle with "Why a Floppy-ROM™ or platter?"

Low cost software distribution is a recurring problem. It just appears that "low cost" and "high reliability" just don't seem to mix. Shipping programs in ROM, at several dollars per copy, is too expensive. All magnetic media cost too much, are fairly bulky, and are fragile. Machine readable print requires a fair amount of hardware.

The bar code reader that we have located costs well over $100. Most people do not have a bar code reader, nor even access to one, but many do have some sort of record player. A conventional record costs a fair amount, and is relatively fragile. The "sound Sheet" or floppy-ROM is low in cost, and best of all, requires little more than a record player and your cassette interface. (Use the card at the end of this book to order the Floppy ROM™.)

What's On The Record?

There are two different sections on the record. First are the test patterns to align your cassette interface. The second section is the software you will want to load into your 6800 based microcomputer. This section of the record includes a binary loader program, and the binary dump of BASIC.

What Will I Need To Use The Record?

A 6800 based microprocessor system is a must. A minimum amount of memory considered should be 6K. In addition, a Kansas City Cassette interface is also a must, and some sort of 300-baud terminal is desirable to check the alignment of your cassette interface. A cassette machine should be used to save BASIC once it gets loaded, because the record will have limited life. Don't rely on it for more than ten or twenty loads because it will wear out. BASIC assumes that MIKBUG is resident in your system, with a scratchpad from A000 to A07F. Patching instructions are provided in an appendix to this article for those systems that do not have MIKBUG and the scratchpad. A separate loader program is also supplied for those without a MIKBUG loader. Several SWTPC 6800 systems have been used by various individuals in verifying the concept behind BASIC on a "Floopy-ROM™," so owners of those systems should not have trouble at all. Last but certainly not least, a 33⅓ RPM turntable.

What Is The "Kansas City Standard?"

The "Kansas City" recording standard was established to allow interchange of data and programs between microcomputer users. It allows great latitude in playback speed, and can be used with almost any cassette machine. "Marks", or logic ones, are represented by a frequency of 2400 Hz, and "spaces", or logic zeros, are represented by 1200 Hz. Data is transferred using the standard UART format, (1 start bit, 1 stop bit) at 300 baud. (Roughly thirty characters per second.) These frequencies were chosen because the 16X clock for the UART can be easily derived from the data being fed into the interface. Speed tolerances of 20- to 30% are acceptable. Figure 1 shows the circuit of the Southwest Technical Products AC-30. It is a fairly simple circuit, which will allow entry of the program into your system. With this circuit, or its equivalent, the program on the "floppy-ROM" may be read.

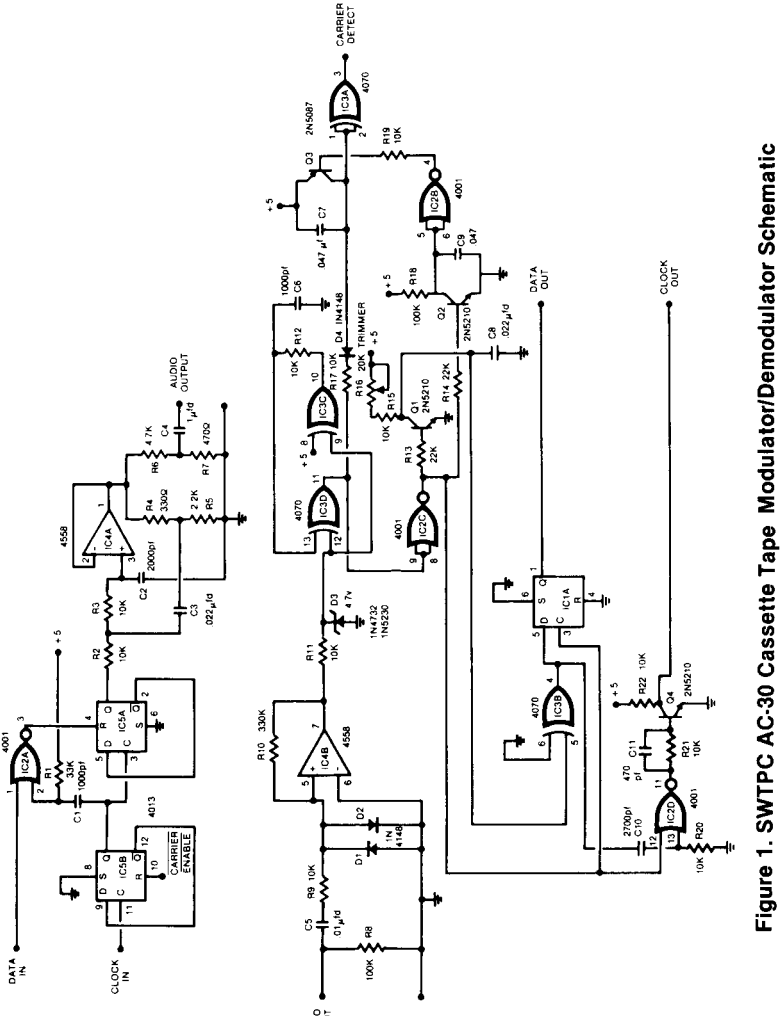


Figure 1. SWTPC AC-30 Cassette Tape Modulator/Demodulator Schematic

How Does It Work?

Audio is fed through the highpass filter made up of R9 and C5. It is clipped by the pair of diodes, and fed into IC-4B, which acts as a comparator. IC-4 should be fed a well regulated 7.5V supply, to ensure stable operation at this point. Zener diode D3 limits the output of IC-4 to levels acceptable to the CMOS gate that follows. When the comparator changes state, IC-3C and IC-3D generate a short (5 microsecond) negative pulse. When data is being received, these pulses repeat. This train of negative pulses grounds C7 through D4. The output of C7 is inverted, buffered, and can be used to generate a carrier detect signal. These negative pulses are found at pin 11 of IC-3D. They're inverted by IC-2A, where they feed four circuits. The first circuit is a missing pulse detector, made up of Q2 and IC-2B. They pull the Carrier detect signal low if several cycles of audio are lost.

The second circuit is an adjustable missing pulse detector, which "times out" when 1200 Hz data are received. R16 trims this time period. The third circuit is flip-flop IC-1A, which outputs the data. Its output is high when 1200 Hz is received. The last circuit is the clock synthesizer, which is made up of IC-2D and IC-3B. These generate the 16X clock for your UART.

How Do I Hook Up A Turntable?

The AC-30 requires about 5 volts of signal to decode the data on the floppy record. Speaker output jacks will easily supply this level. Before you hook up a phonograph, check to see if it has a transformer-isolated chassis. If it doesn't, *don't hook it up!* It would be all too easy to get 120V of your local power utility into your computer. This would tend to make a very expensive pile of write-only memory. If necessary, have a technician check out your stereo. Some component pre-amps are capable of driving 5 volts. A Dyna Pat-4 was used to test the records, and that worked nicely. However, it had to be turned up all the way. If your stereo set has a MONO—STEREO switch, set it on mono. A large percentage of noise on this type of disc is vertical. The vertical signal is essentially a "difference" signal for stereo, but is unwanted for this playback. It will cancel nicely, which will help load a very noisy disc.

Adjust your stereo set for 'flat' response. If your tone control is a 'High Cut' type, set the control for maximum. This is the starting point to align your system. If possible, transcribe the record the first time it is played. It should have reasonable life, 20 to 30 plays or more, but better safe than sorry.

What Are The Weird Signals At The Start Of The Record?

There are two test patterns that should be used to set up your AC-30. The first is a pattern of "5"s. Set your system up to echo the data from the tape to your terminal (local mode). Play back the tape. You should see a stream of "5"s. If you don't, adjust the tone and volume controls slowly and carefully. The "5"s should play back

perfectly. There should be no other characters intermixed with the "5"s.

If you are not able to get the "5"s to play back properly, check your interface circuitry next. If you built the interface shown in Figure 1, adjust the R16 to trim the test pattern.

The second test pattern is a "U*" pattern, to check for jitter. If the "5"s worked properly, this should also. If you are not able to get the "5"s to read out, don't go further. Examine your connections, make sure you do not have any ground loops. Make sure you are getting a good clean signal out of the turntable. If you re-recorded the record on a cassette, make sure that you did not overload the cassette recorder when recording. Figure 2, shows a general view of how your system should be hooked up, with the turntable hooked into the AC-30 in place of a cassette machine. If you have arrived this far with flying colors, now is the time to load the program. First check your memory. Make sure all is well before going further.

A slightly high boost may be necessary in some systems. If your system has a rumble filter, switch it on. There will be no valid information below 100 Hz, and elimination of these noise components may be helpful.

If your record player has a good "scratch" filter, try it too. There is no information above 2400, but response at 2400 should be left as "Flat" as possible. If your scratch filter affects 5000 and above, it may be OK; 10000 and above would be better. Remember that most filters have a finite slope and frequencies 2 to 3 octaves away from the break frequency will be affected by level and phase changes. If you detect errors during program load, turn off the 'scratch' filter first.

What If My Turntable Runs Fast Or Slowly?

Again, the beauty of the "Kansas City" standard comes to the rescue. Very few turntables have speed errors larger than 10%, so there is little reason for concern. If your error is very large, adjust your cassette interface card. Run as close to 33⅓ as possible on an adjustable speed turntable.

How To Load The Program

After the test patterns, there is a 20-second space where the 'band' is cut. A string of 'L's follow, to start the loader function in MIKBUG. Place the tone arm down on the record near the end of the test patterns. There is a binary loader program in front of the data, which MIKBUG loads into memory at 1100 HEX. The computer than executes the binary load program, and loads BASIC. Very simple if you have MIKBUG. Figure 4, is a listing of the loader program which was loaded at 1100 HEX. You will note several MIKBUG routines were used. Patch these to fit your operating system. If your turntable has a center depression, place a regular record on the turntable first to support the Floppy-ROM™.

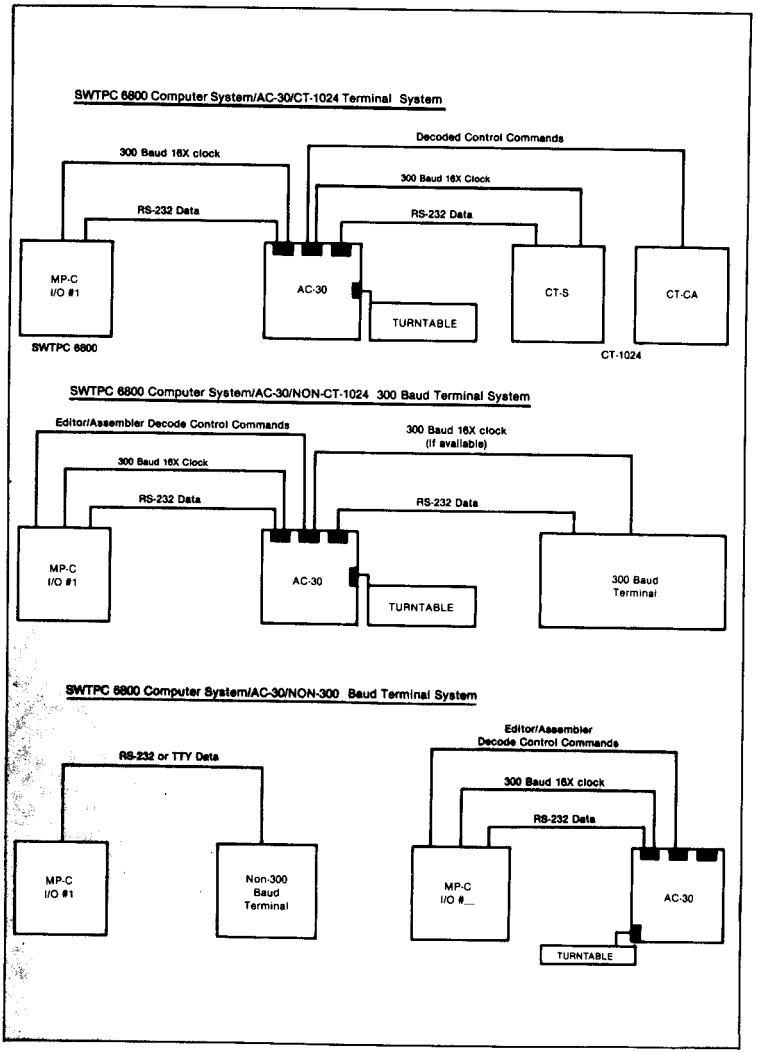


Figure 2. Interconnection Block Diagrams

Pointers And External Routines

Several subroutines in MIKBUG® were used. Copies of these routines may be found in Motorola Engineering Note #100, which is a listing of MIKBUG®. The routines are shown in Listing 1. Here is a short summary of the external subroutines, and a short explanation of what they do.

```
0272  7E  E0BF  OUT2H   JMP   $E0BF   OUT2H IN MIKBUG
0275  7E  E0C8  OUT4HS  JMP   $E0C8   OUT4HS IN MIKBUG
0278  8D  09    OUTCH   BSR   BREAK
027A  7E  E1D1          JMP   $E1D1   OUTEEE IN MIKBUG
027D  BD  E1AC  INCH    JSR   $E1AC   INEEE IN MIKBUG
0280  36                PSH A
0281  20  08            BRA   BREAK0
0283  36        BREAK   PSH A
0284  B6  8004          LDA A $8004
0287  2B  09            BMI   BREAK1
0289  8D  F2            BSR   INCH
028B  81  03    BREAK0  CMP A #$03
028D  26  03            BNE   BREAK1
028F  7E  0815          JMP   READY   NOTE READY LOCATION
0292  32        BREAK1  PUL A
0293  39                RTS           DONE
```

Listing 1.

The first of these routines is "OUT2H." This routine prints the contents of memory pointed to by the index register. The contents are printed as two HEX characters. The index register is incremented before returning. A comparable routine should be found in almost all operating systems. The jump location is:

0272 (HEX): 7E, E0, BF

EOBF is the location of OUT2H in MIKBUG®, 7E is the jump instruction.

The second external routine called is "OUT4HS." This routine will print 4 HEX characters by calling the OUT2H routine twice. It then outputs a space. Again, there is an equivalent in almost all operating systems. This routine returns the Index Register incremented twice. The instruction is located at:

0275: 7E, E0, C8

EOCB is the address of OUT4HS in MIKBUG®.

The input and output routines in MIKBUG® are also used. INEEE and OUTEEE transfer one character through 'A' when called. It should be noted that the control port on a MIKBUG® equipped system is a 'bitbanger' and thus may have different status bits than an ACIA. The jump locations are:

INEEE:

027D: BD, E1, AC (Note it is a jump to subroutine)

OUTEEE:

027A: 7E, E1, D1 (Note that this is a jump)

Listing 1 shows the block of code containing these calls. The listing also has a listing of the break routine. If you are using an ACIA, it will

be necessary to change the break routine to check for your status. You will probably have to add a short subroutine to the end of BASIC.

Another not-so-critical patch location is the carriage return-line feed string. The 15(HEX) is the erase to end of line character on a SWTPC CT-1024 terminal. This may be changed as necessary to allow an erase on your terminal. The string is located at:

02AC: 0D, 0A, 15, FF, FF, FF, FF, 1E

If you have to add a routine to the end of BASIC, the pointer for the beginning of program storage must be changed, or BASIC will dramatically change your code! The locations you must change are:

07F9, 07FA

They currently contain 1200 HEX. You would have to change the contents of these locations to point at a location after your routine(s). Any routines that you have to add, should be located beyond 1200.

Stack load and store operating may have to be changed if you don't have memory from A000 to A07F. There are four stack pointer instructions of note. These all load the stack pointer with A045 HEX. These locations are:

080F, 0848, 0B16, 0CC3

There is also a pair of stack references in the patch command routine. The first is at 08F8. This contains A040 currently. The instruction is located in 08FB then stores the stack in A008. Note...the "Patch" command has another jump that may have to be changed if you don't have MIKBUG®. 08E3 contains the jump to MIKBUG®. It contains E0E3 currently. Change this to the equivalent in your monitor.

There is an index register stack which has one reference. The index register stack sits at the top of MIKBUG's scratchpad. Location 0819 contains the Value A07F. Move this elsewhere as necessary.

There is a store index register command in the "Patch" routine. It is located at 08F5, and contains an A046. This location is the top of the stack, so the 'G' command will cause a return to BASIC.

Listing 2 shows a possible break routine for those with an ACIA for input. Note that it will not fit in the room that the current break routine occupies. Change the current break to have a jump subroutine at 1200 HEX. A return would follow that.

Listing 3 is a command listing of the biload program. It may be 'patched' to fit into most operating systems, and will be quite easy to implement on a system using an ACIA. The data is transferred through the A accumulator. In MITS 680b systems, you will have to transfer from B to A because they use B for input and output. The place to put your input one character routine call is INCHP. Load your BILOAD equivalent, then execute it. It will ignore the BILOAD program on the FLOPPY ROM™ and load BASIC (hopefully).

```
00001                          NAM       BREAK
00002                          OPT       S,P
00003        0815    READY     EQU       $0815     INSIDE BASIC....
00004        E000    PORT      EQU       $E000
00005 1200                     ORG       $1200
00006 1200 36        BREAK     PSH A
00007 1201 B6 E000             LDA A     PORT      (THIS IS THE STATUS PORT)
00008 1204 48                  ASL A               PUT DATA AVAILABLE FLAG IN
00009 1205 24 0C               BCC       RETURN    NOTHING THERE..BRANCH
00010 1207 B6 E001             LDA A     PORT+1    BRING IN A CHAR IF THERE
00012 120A 84 7F               AND A     #$7F      GET RID OF PARITY...
00013 120C 81 03               CMP A     #$03      IS IT A BREAK CHARACTER?
00014 120E 26 03               BNE       RETURN    NO...
00015 1210 7E 0815             JMP       READY     YES IT IS!! STOP!
00016                  *
00017 1213 32        RETURN    PUL A
00018 1214 39                  RTS                 GO BACK AGAIN...
00019                          END

READY  0815
PORT   E000
BREAK  1200
RETURN 1213
```

Listing 2.

```
00030                          NAM       BILOAD
02780                          OPT       L
02790 19A4 8E 1500 BILOAD      LDS       #$1500    NEW STACK LOCATION
02800 19A7 8D 49               BSR       LOAD      GET IO READY
02810 19A9 8D 3C    OVER       BSR       INPUT     GET A CHARACTER
02820 19AB 81 58               CMP A     #'X       IS IT FIRST CHARACTER?
02830 19AD 26 FA               BNE       OVER      IF NOT, KEEP LOOKING.....
02840 19AF 8D 36               BSR       INPUT     GET ANOTHER CHARACTER
02850 19B1 81 31               CMP A     #'1       SECOND CHARACTER?
02860 19B3 27 07               BEQ       READ      GET DATA AND MORE...
02870 19B5 81 39               CMP A     #'9       END CHARACTER?
02880 19B7 26 F0               BNE       OVER      KEEP LOOKING
02890 19B9 7E E0E3             JMP       CONTRL    PATCH HERE FOR OTHER
02895                  2895 *                        6800 SYSTEMS

02910 19BC 7F 17FF READ        CLR       CKSM      START WITH ZERO
02920 19BF 8D 26               BSR       INPUT     GET BYTE COUNT
02930 19C1 16                  TAB                 PUT IT IN BP
02940 19C2 5C                  INC B     *         INCREMENT IT ONCE
02950 19C3 8D 22               BSR       INPUT     GET FIRST PART OF ADDRESS
02960 19C5 B7 1800             STA A     TW        STORE IT TEMPORARILY
02970 19C8 8D 1D               BSR       INPUT     GET SECOND PART
02980 19CA B7 1801             STA A     TW+1      STORE IT
02990 19CD FE 1800             LDX       TW        PUT IT ALL IN INDEX

03010 19D0 8D 15    STORE      BSR       INPUT     GET SOME DATA
03020 19D2 A7 00               STA A     X         STORE IT
03030 19D4 01                  NOP                 TIME TO FORGET..

03050 19D5 A1 00               CMP A     X         IF SAME, GO ON
03060 19D7 26 0B               BNE       OUT       IF NOT, NO MEMORY THERE
03070 19D9 08                  INX                 POINT AT NEXT LOCATION

03090 19DA 5A                  DEC B     *         SUBTRACT 1 BYTE FROM COUNT
03100 19DB 26 F3               BNE       STORE     NOT DONE, DO MORE
03110 19DD 8D 08               BSR       INPUT     OK, ADD IN TAPE CHECKSUM
03120 19DF 7C 17FF             INC       CKSM      ADD ONE...
03130 19E2 27 C5               BEQ       OVER      IF ZERO, ALL IS GOOD
03140 19E4 7E E040 OUT         JMP       LOAD19    AN ERROR! PRINT GARBAGE
03150 19E7 8D 14    INPUT      BSR       INCHP     INPUT 1-8 BIT CHARACTER
03160 19E9 36                  PSH A     *         SAVE IT
03170 19EA BB 17FF             ADD A     CKSM      ADD SOME TO CHECKSUM
03180 19ED B7 17FF             STA A     CKSM      STORE THE RESULTS
03190 19F0 32                  PUL A     *         RECALL THE DATA
03200 19F1 39                  RTS                 GO BACK
```

```
03220 19F2 86 11   LOAD   LDA A  #$11      START YOUR ENGINES
03230 19F4 BD E1D1        JSR    OUTEEE    TELL THE MACHINE TO GO
03240 19F7 86 3C          LDA A  #$3C      PIA READER CONTROL
03250 19F9 B7 8007        STA A  $8007     PUT IT INTO PIA
03260 19FC 39             RTS

03271              3271 * NOTE...IF YOU HAVE A ROUTINE THAT WILL
03272              3272 * INPUT A FULL 8 BIT WORD WITH PARITY
03273              3273 * CALL YOUR ROUTINE IN PLACE OF INCHP...
03274              3274 * FOR EXAMPLE...
03275              3275 *  INCHP PSH B       PROTECT B
03276              3276 *        JSR INCHAR  THIS IS YOUR ROUTINE

---
03277              3277 *        PUL B       GET B BACK
03278              3278 *        RTS
03279              3279 * PROTECT INDEX REGISTER AT ALL TIMES!!!
03280 19FD 37     INCHP  PSH B  *           THIS IS ALMOST SAME
03290 19FE BD E1A5        JSR    SAV       AS MIKBUG,BUT SAVES PARITY
03300 1A01 A6 00   IN1    LDA A  X         BIT. SEE MOTOROLA ENG.LOG
03310 1A03 2B FC          BMI    IN1       FOR DETAILS OF HOW IT WORKS

03330 1A05 6F 02          CLR    2,X
03340 1A07 BD E1F3        JSR    DE
03350 1A0A BD E1EF        JSR    DEL
03360 1A0D C6 04          LDA B  #4
03370 1A0F E7 02          STA B  2,X
03380 1A11 58             ASL B

03400 1A12 BD E1EF IN3    JSR    DEL
03410 1A15 0D             SEC
03420 1A16 69 00          ROL    X
03430 1A18 46             ROR A
03440 1A19 5A             DEC B
03450 1A1A 26 F6          BNE    IN3
03460 1A1C BD E1EF        JSR    DEL
03470 1A1F 7E E1E3        JMP    IOUT2
```

Listing 3.

Memory Map for 4K Basic

0000-00FF	Input buffer and temporary variable storage
0100	Hard starting address of BASIC (Cold start—clears program)
0103	Soft starting address of BASIC (warm start—saves program)
0105-10FF	BASIC Interpreter
1100-11FF	Arithmetic and For-Next stack
A000-A045	Machine stack
A04A-A07F	Index register stack

Note: Binary loader program starts at 1100. It will be cleared by the Basic interpreter when 0100 is executed.

```
02775 1100                ORG    $1100
02780 1100 8E A047 BILOAD LDS    #$A047
02790 1103 8D 49          BSR    LOAD

02800 1105 8D 3C   OVER   BSR    INPUT
02810 1107 81 58          CMP A  #'X
02820 1109 26 FA          BNE    OVER
02830 110B 8D 36          BSR    INPUT
02840 110D 81 31          CMP A  #'1
02850 110F 27 07          BEQ    READ
02860 1111 81 39          CMP A  #'9
```

```
02870 1113 26 F0          BNE     OVER
02880 1115 7E E0E3        JMP     CONTRL


02900 1118 7F A016 READ   CLR     CKSM
02910 111B 8D 26          BSR     INPUT
02920 111D 16             TAB
02930 111E 5C             INC B
02940 111F 8D 22          BSR     INPUT
02950 1121 B7 A019        STA A   TW
02960 1124 8D 1D          BSR     INPUT
02970 1126 B7 A01A        STA A   TW+1
02980 1129 FE A019        LDX     TW


03000 112C 8D 15   STORE   BSR     INPUT
03010 112E A7 00          STA A   X
03020 1130 01             NOP
03040 1131 A1 00          CMP A   X
03050 1133 26 0B          BNE     OUT
03060 1135 08             INX
03080 1136 5A             DEC B
03090 1137 26 F3          BNE     STORE
03100 1139 8D 08          BSR     INPUT
03110 113B 7C A016        INC     CKSM
03120 113E 27 C5          BEQ     OVER


03130 1140 7E E040 OUT    JMP     LOAD19

03140 1143 8D 14   INPUT  BSR     INCHP
03150 1145 36             PSH A
03160 1146 BB A016        ADD A   CKSM
03170 1149 B7 A016        STA A   CKSM
03180 114C 32             PUL A
03190 114D 39             RTS


03210 114E 86 11   LOAD   LDA A   #$11
03220 1150 BD E1D1        JSR     OUTEEE
03230 1153 86 3C          LDA A   #$3C
03240 1155 B7 8007        STA A   $8007
03250 1158 39             RTS
03270 1159 37      INCHP  PSH B
03280 115A BD E1A5        JSR     SAV

03290 115D A6 00   IN1     LDA A   X
03300 115F 2B FC          BMI     IN1
03320 1161 6F 02          CLR     2,X
03330 1163 BD E1F3        JSR     DE
03340 1166 BD E1EF        JSR     DEL
03350 1169 C6 04          LDA B   #4
03360 116B E7 02          STA B   2,X
03370 116D 58             ASL B


03390 116E BD E1EF IN3    JSR     DEL
03400 1171 0D             SEC
03410 1172 69 00          ROL     X
03420 1174 46             ROR A
03430 1175 5A             DEC B
03440 1176 26 F6          BNE     IN3


03450 1178 BD E1EF        JSR     DEL
03460 117B 7E E1E3        JMP     IOUT2
```

Listing 4.

```
                        NAM  BASIC 4K
                        *
                        *
                        ******* VERSION 2.1 *******
                        *
                        *    BY ROBERT H. UITERWYK
                        *    PLACED IN THE PUBLIC
                        *    DOMAIN    JUNE, 1979 FOR
                        *    THE PURPOSES OF EDUCATION
                        *    FOR ALL MICROCOMPUTER
                        *    USERS.
                        *
                        * THIS PROGRAM ASSUMES THAT THE
                        * MOTOROLA MIKBUG ROM IS INSTALLED
                        * AND THAT ITS ASSOCIATED 128 BYTE
                        * RAM IS ALSO PRESENT
                        * THE SP AND XSTACK WILL HAVE TO
                        * BE MOVED IF THIS IS NOT THE CASE.


00270                   *SAVE FIRST $20 BYTES FOR
00280                   *FUTURE DISK USE
00290 0020              ORG    $20
00300 0020 0002 INDEX1  RMB    2
00310 0022 0002 INDEX2  RMB    2
00320 0024 0002 INDEX3  RMB    2
00330 0026 0002 INDEX4  RMB    2
00340 0028 0002 SAVESP  RMB    2
00350 002A 0002 NEXTRA  RMB    2
00360 002C 0002 WORKBA  RMB    2
00370 002E 0002 SOURCE  RMB    2
00380                   *TO BE CHANGED AT PROGRAMMERS OPTION
00390 0030 0002 PACKLN  RMB    2
00400 0032 0002 HIGHLN  RMB    2
00410 0034 0002 BASPNT  RMB    2
00420 0036 0002 BASLIN  RMB    2
00430 0038 0002 PUSHTX  RMB    2
00440 003A 0002 XSTACK  RMB    2
00450 003C 0002 DATPNT  RMB    2
00460 003E 0002 DIMPNT  RMB    2
00470 0040 0001 PRCNT   RMB    1
00480 0041 0003         RMB    3
00490 0044 0002 MEMEND  RMB    2
00510 0046 0002 ARRTAB  RMB    2
00520 0048 0002 KEYWD   RMB    2
00530 004A 0001 TSIGN   RMB    1
00540 004B 0001 EXP     RMB    1
00550 004C 0001 SIGDIG  RMB    1
00560 004D 0001 DECFLG  RMB    1
00570 004E 0001 TOGGLE  RMB    1
00580 004F 0001 NCMPR   RMB    1
00590 0050 0001 TNUMR   RMB    1
00600 0051 0001 ANUMR   RMB    1
00610 0052 0001 BNUMR   RMB    1
00620 0053 0001 LRNINF  RMB    1
00630 0054 0001 ROUNDV  RMB    1
00640 0055 0001 ONLOOP  RMB    1
00650 0056 0001 NORSGN  RMB    1
00660 0057 0002 XLONG   RMB    2
00670 0059 0002 MPYXX   RMB    2
00680 005B 0002 BASNUM  RMB    2
00690 005D 0002 AFSTK   RMB    2
00700 005F 0002 FORPNT  RMB    2
00710 0061 0002 FORNOW  RMB    2
00720 0063 0002 VARPNT  RMB    2
00730 0065 0002 SBRPNT  RMB    2
00740 0067 0002 DUMMY   RMB    2
00750 0069 0006 RANNUM  RMB    6
00760 006F 0006 FORTMP  RMB    6
00770 0075 0010 SBRSTK  RMB    16
00780 0085 0001 LOADFLG RMB    1


00800 00AC              ORG    $00AC
00810 00AC 0002 BUFNXT  RMB    2
00820 00AE 0002 ENDBUF  RMB    2
00830 0080              ORG    $0080
00840 0080 0050 BUFFER  RMB    $50      I/O BUFFER = 80 DECIMAL
```

```
00860 0100                      ORG     $0100
00870 0100 BD 07F2 PROGM        JSR     START
00880 0103 7F 07F6              JMP     READY
00890 0106 54      INTER        FCC     /TO/
      0107 4F
00900 0108 1F                   FCB     $1F
00910 0109 9999    TOPNT        FDB     $9999
00920 010B 53                   FCC     /STEP/
      010C 54
      010D 45
      010E 50
00930 010F 1F                   FCB     $1F
00940 0110 8888    STEPNT       FDB     $8888
00950 0112 52      COMMAN       FCC     /RUN/
      0113 55
      0114 4F
00960 0115 1F                   FCB     $1F
00970 0116 085B                 FDB     RUN
00980 0118 4C                   FCC     /LIST/
      0119 49
      011A 53
      011B 54
00990 011C 1F                   FCB     $1F
01000 011D 0882                 FDB     CLIST
01010 011F 4F                   FCC     /NEW/
      0120 45
      0121 57
01020 0122 1F                   FCB     $1F
01030 0123 07F2                 FDB     START
01040 0125 50                   FCC     /PAT/
      0126 41
      0127 54
01050 0128 1F                   FCB     $1F
01060 0129 08D3                 FDB     PATCH
01070 012B 4C                   FCC     /LOA/
      012C 4F
      012D 41
01080 012E 1F                   FCB     $1F
01090 012F 090D                 FDB     LOAD
01100 0131 53                   FCC     /SAV/
      0132 41
      0133 56
01110 0134 1F                   FCB     $1F
01120 0135 08E5                 FDB     SAVE
01130 0137 47      GOLIST       FCC     /GOSUB/
      0138 4F
      0139 53
      013A 55
      013B 42
01140 013C 1F                   FCB     $1F
01150 013D 0A5A                 FDB     GOSUB
01160 013F 47                   FCC     /GOTO/
      0140 4F
      0141 54
      0142 4F
01170 0143 1F                   FCB     $1F
01180 0144 0A7C                 FDB     GOTO
01190 0146 4F                   FCC     /ON/
      0147 4E
01200 0148 1F                   FCB     $1F
01210 0149 0A3B                 FDB     ONGO
01220 014B 54                   FCC     /THEN/
      014C 48
      014D 45
      014E 4E
01230 014F 1F                   FCB     $1F
01240 0150 0A29                 FDB     THEN
```

```
01250 0152 50              FCC     /PRINT/
      0153 52
      0154 49
      0155 4F
      0156 54
01260 0157 1F              FCB     $1F
01270 0158 0C6D            FDB     PRINT
01280 015A 4C              FCC     /LET/
      015B 45
      015C 54
01290 015D 1F              FCB     $1F
01300 015F 0DBC            FDB     LET
01310 0160 49              FCC     /INPUT/
      0161 4F
      0162 50
      0163 55
      0164 54
01320 0165 1F              FCB     $1F
01330 0166 0AC4            FDB     INPUT
01340 0168 49              FCC     /IF/
      0169 46
01350 016A 1F              FCB     $1F
01360 016B 0FDD            FDB     IF
01370 016D 52              FCC     /READ/
      016E 45
      016F 41
      0170 44
01380 0171 1F              FCB     $1F
01390 0172 0C17            FDB     READ
01400 0174 44              FCC     /DATA/
      0175 41
      0176 54
      0177 41
01410 0178 1F              FCB     $1F
01420 0179 0DDF            FDB     REMARK
01430 017B 52              FCC     /RESTORE/
      017C 45
      017D 53
      017E 54
      017F 4F
      0180 52
      0181 45
01440 0182 1F              FCB     $1F
01450 0183 0C64            FDB     RESTORE
01460 0185 45              FCC     /END/
      0186 4E
      0187 44
01470 0188 1F              FCB     $1F
01480 0189 07F6            FDB     READY
01490 018B 52              FCC     /RETURN/
      018C 45
      018D 54
      018E 55
      018F 52
      0190 4E
01500 0191 1F              FCB     $1F
01510 0192 0AA2            FDB     RETURN
01520 0194 44              FCC     /DIM/
      0195 49
      0196 4D
01530 0197 1F              FCB     $1F
01540 0198 0F06            FDB     DIM
01550 019A 46              FCC     /FOR/
      019B 4F
      019C 52
01560 019D 1F              FCB     $1F
01570 019E 0F86            FDB     FOR
01580 01A0 4E              FCC     /NEXT/
      01A1 45
```

```
      01A2 58
      01A3 54
01590 01A4 1F             FCB   $1F
01600 01A5 0F50           FDB   NEXT
01610 01A7 52             FCC   /REM/
      01A8 45
      01A9 4D
01620 01AA 1F             FCB   $1F
01630 01AB 0DDF           FDB   REMARK
01640 01AD 41             FCC   /APP/
      01AE 50
      01AF 50
01650 01B0 1F             FCB   $1F
01660 01B1 0910           FDB   APPEND
01670 01B3 53       STOMSG FCC  /STOP/
      01B4 54
      01B5 4F
      01B6 50
01680 01B7 1F             FCB   $1F
01690 01B8 0AB6           FDB   STOP
01700 01BA 1F       COMEND FCB  $1F
01710 01BB 0DBC     IMPLET FDB  LET



01730 01BD 52       FUNCT FCC   /RND(/
      01BE 4E
      01BF 44
      01C0 28
01740 01C1 1F             FCB   $1F
01750 01C2 10B7           FDB   RANDOM
01760 01C4 54             FCC   /TAB(/
      01C5 41
      01C6 42
      01C7 28
01770 01C8 1F             FCB   $1F
01780 01C9 0CB2           FDB   TAB
01790 01CB 49             FCC   /INT(/
      01CC 4E
      01CD 54
      01CE 28
01800 01CF 1F             FCB   $1F
01810 01D0 1079           FDB   INT
01820 01D2 43             FCC   /CHR$(/
      01D3 48
      01D4 52
      01D5 24
      01D6 28
01830 01D7 1F             FCB   $1F
01840 01D8 0CA5           FDB   CHR
01850 01DA 41             FCC   /ABS(/
      01DB 42
      01DC 53
      01DD 28
01860 01DE 1F             FCB   $1F
01870 01DF 10AE           FDB   ABS
01880 01E1 53             FCC   /SGN(/
      01E2 47
      01E3 4E
      01E4 28
01890 01E5 1F             FCB   $1F
01900 01E6 1093           FDB   SGN
01910 01E8 55             FCC   /USER(/
      01E9 53
      01EA 45
      01EB 52
      01EC 28
```

```
01920 01ED 1F       FUNEND FCB    $1F
01930 01EE 106R            FDB    USER
01950 01F0 13       RDYMSG FCB    $13
01960 01F1 14              FCB    $14
01970 01F2 0D              FCB    $0D
01980 01F3 0A              FCB    $0A
01990 01F4 15              FCB    $15
02000 01F5 52              FCC    /READY/
      01F6 45
      01F7 41
      01F8 44
      01F9 59
02010 01FA 1F              FCB    $1F
02020 01FB 20       DELMSG FCC    / DEL/
      01FC 44
      01FD 45
      01FE 4C
02030 01FF 1F              FCB    $1F
02040 0200 10       PGCNTL FCB    $10
02050 0201 16              FCB    $16
02060 0202 1F              FCB    $1F
02070 0203 52       WHAT   FCC    /RE-ENTER/
      0204 45
      0205 2D
      0206 45
      0207 4E
      0208 54
      0209 45
      020A 52
02080 020B 1E              FCB    $1E
02090 020C 45       ERRMS1 FCC    /ERROR# /
      020D 52
      020E 52
      020F 4F
      0210 52
      0211 23
      0212 20
02100 0213 1F              FCB    $1F
02110 0214 20       ERRMS2 FCC    / IN LINE /
      0215 49
      0216 4E
      0217 20
      0218 4C
      0219 49
      021A 4E
      021B 45
      021C 20
02120 021D 1F              FCB    $1F
02140 021E CE 01FB DEL     LDX    #DELMSG
02150 0221 8D 6E           BSR    OUTPUT
02160 0223 20 07           BRA    KEYBDO
02170                *ALL PURPOSE KEYBD ROUTINE
02180 0225 86 3F    KEYBD  LDA A  #$3F      OUTPUT "?"
02190 0227 8D 4D           BSR    OUTCH
02200 0229 BD 0350         JSR    PRINSP
02210                *SUBROUTINE ENTRY WITHOUT "?"
02220 022C CE 00B0 KEYBDO  LDX    #BUFFER    START OF KEYBD ROUTINE
02230 022F 8D 4A    KEYBD1 BSR    INCH       GET A CHARACTER
02240 0231 81 18           CMP A  #$18
02250 0233 27 F9           BEQ    DEL
02260 0235 81 0D           CMP A  #$0D       IS IT C/R
02270 0237 27 2A           BEQ    TEXIT      IF SO EXIT
02280 0239 81 0F           CMP A  #$0F
02290 023B 26 0C           BNE    *+14
02300 023D 86 5F           LDA A  #$5F
02310 023F 8D 35           BSR    OUTCH
02320 0241 8C 00B0         CPX    #BUFFER    IS IT FIRST CHARACTER
02330 0244 27 E9           BEQ    KEYBD1     IF SO, JUST RELOOP
02340 0246 09              DEX               OTHERWISE, BACKSPACE
```

```
02350 0247 20 F6          BRA     KEYBD1    AND THEN RELOOP
02360 0249 7D 0085        TST     LOAFLG
02370 024C 26 0A          BNE     KEYBD2
02380 024E 8C 00F8        CPX     #BUFFER+72
02390 0251 26 05          BNE     *+7
02400 0253 C6 21          LDA B   #$21
02410 0255 7E 0828        JMP     ERROR
02420 0258 A7 00    KEYBD2 STA A   0,X
02430 025A 08             INX
02440 025B 20 D2          BRA     KEYBD1
02450 025D 86 23    CNTLIN LDA A   #'#
02460 025F 8D 15          BSR     OUTCH
02470 0261 20 C9          BRA     KEYBD0
02480 0263 86 1E    TEXIT  LDA A   #$1E
02490 0265 A7 00          STA A   X
02500 0267 DF AE          STX     ENDBUF
02510 0269 96 85          LDA A   LOAFLG
02520 026B 26 02          BNE     *+4
02530 026D 8D 31          BSR     CRLF
02540 026F 39             RTS


02560                ***********************************************
02570                *
02580                *     THIS PAGE CONTAINS EXTERNAL ROUTINE CALLS
02590                *
02600                ***********************************************
02610 0270 7E F0BF OUT2H  JMP     $F0BF     OUT2H IN MIKBUG


02630 0273 7E F0C8 OUT4HS JMP     $F0C8     OUT4HS IN MIKBUG


02650 0276 8D 09   OUTCH  BSR     BREAK
02660 0278 7E F1D1        JMP     $F1D1     OUTEEE IN MIKBUG

02680 027B BD F1AC INCH   JSR     $F1AC     INEEE IN MIKBUG
02690 027E 36             PSH A
02700 027F 20 08          BRA     BREAK0

02720 0281 36       BREAK PSH A
02730 0282 B6 8004        LDA A   $8004
02740 0285 2B 09          BMI     BREAK1
02750 0287 8D F2          BSR     INCH
02760 0289 81 03    BREAK0 CMP A  #$03
02770 028B 26 03          BNE     BREAK1
02780 028D 7E 07F6        JMP     READY
02790 0290 32       BREAK1 PUL A
02800 0291 39             RTS
02830                *UNIVERSAL OUTPUT ROUTINE
02840       0292    OUTPUT EQU    *
02850 0292 8D 05          BSR     OUTNCR
02860 0294 20 0A          BRA     CRLF
02870 0296 8D DE          BSR     OUTCH
02880 0298 08             INX
02890 0299 A6 00    OUTNCR LDA A  0,X
02900 029B 81 1F          CMP A   #$1F
02910 029D 26 F7          BNE     *-7
02920 029F 39             RTS
02930 02A0 8D 10    CRLF  BSR     PUSHX
02940 02A2 CE 02AA        LDX     #CRLFST
02950 02A5 8D F2          BSR     OUTNCR
02960 02A7 8D 1F          BSR     PULLX
02970 02A9 39             RTS
02980 02AA 0D       CRLFST FCB    $0D
02990 02AB 0A             FCB     $0A
03000 02AC 15             FCB     $15
03010 02AD FF             FCB     $FF,$FF,$FF,$FF
      02AE FF
```

```
      02AF FF
      02B0 FF
03020 02B1 1F       CREND   FCB     $1F


03040 02B2 DF 38    PUSHX   STX     PUSHTX
03050 02B4 DE 3A            LDX     XSTACK
03060 02B6 09               DEX
03070 02B7 09               DEX
03080 02B8 DF 3A            STX     XSTACK
03090 02BA 36               PSH A
03100 02BB 96 38            LDA A   PUSHTX
03110 02BD A7 00            STA A   0,X
03120 02BF 96 39            LDA A   PUSHTX+1
03130 02C1 A7 01            STA A   1,X
03140 02C3 32               PUL A
03150 02C4 DE 38            LDX     PUSHTX
03160 02C6 39               RTS


03180 02C7 DE 3A    PULLX   LDX     XSTACK
03190 02C9 FF 00            LDX     0,X
03200 02CB 7C 003B          INC     XSTACK+1
03210 02CE 7C 003B          INC     XSTACK+1
03220 02D1 39               RTS




03240 02D2 8D DF    STOREX  BSR     PUSHX
03250 02D4 BD 0448          JSR     PULLAF
03260 02D7 9F 28            STS     SAVESP
03270 02D9 35               TXS
03280 02DA DE 3A            LDX     XSTACK
03290 02DC EE 00            LDX     0,X
03300 02DE 20 11            BRA     *+19
03310 02E0 8D D0    STORE   BSR     PUSHX
03320 02E2 BD 0448          JSR     PULLAF
03330 02E5 9F 28            STS     SAVESP
03340 02E7 35               TXS
03350 02E8 09               DEX
03360 02E9 09               DEX
03370 02EA 09               DEX
03380 02EB 09               DEX
03390 02EC 09               DEX
03400 02ED 09               DEX
03410 02EE 09               DEX
03420 02EF EE 00            LDX     0,X
03430 02F1 C6 05            LDA B   #5
03440 02F3 32               PUL A
03450 02F4 A7 00            STA A   0,X
03460 02F6 08               INX
03470 02F7 5A               DEC B
03480 02F8 26 F9            BNE     *-5
03490 02FA 32               PUL A
03500 02FB 32               PUL A
03510 02FC A7 00            STA A   0,X
03520 02FE 9E 28            LDS     SAVESP
03530 0300 20 C5            BRA     PULLX




03550 0302 8D AE    INDX    BSR     PUSHX
03560 0304 9F 28            STS     SAVESP
03570 0306 35               TXS
03580 0307 20 0A            BRA     INDO
```

```
03590 0309 8D A7    IND     BSR     PUSHX
03600 030B BD 0448          JSR     PULLAE
03610 030E 9F 28            STS     SAVESP
03620 0310 AE 00            LDS     0,X
03630 0312 34               DES
03640 0313 C6 05    INDO    LDA B   #5
03650 0315 DE 5D            LDX     AESTK
03660 0317 32               PUL A
03670 0318 A7 00            STA A   0,X
03680 031A 08               INX
03690 031B 5A               DEC B
03700 031C 26 F9            BNE     *-5
03710 031E 32               PUL A
03720 031F A7 01            STA A   1,X
03730 0321 6F 00            CLR     0,X
03740 0323 9F 28            LDS     SAVESP
03750 0325 BD 043F          JSR     PUSHAE
03760 0328 20 9D            BRA     PULLX
```

```
03780 032A BD 0273 OUTLIN JSR     OUT4HS
03790 032D A6 00          LDA A   0,X
03800 032F 08             INX
03810 0330 BD 02B2        JSR     PUSHX
03820 0333 97 49          STA A   KEYWD+1
03830 0335 DE 48          LDX     KEYWD
03840 0337 09             DEX
03850 0338 09             DEX
03860 0339 A6 00          LDA A   0,X
03870 033B 81 1E          CMP A   #$1E
03880 033D 26 F9          BNE     *-5
03890 033F 08             INX
03900 0340 08             INX
03910 0341 08             INX
03920 0342 BD 0299        JSR     OUTNCR
03930 0345 8D 09          BSR     PRINSP
03940 0347 7F 0040        CLR     PRCNT
03950 034A BD 02C7        JSR     PULLX
03960 034D 7E 0292        JMP     OUTPUT
```

```
03980 0350 36       PRINSP PSH A
03990 0351 86 20           LDA A   #$20
04000 0353 BD 0CF6         JSR     OUTCHA
04010 0356 32              PUL A
04020 0357 39              RTS
```

```
04040 0358 BD 0744 FINVAR JSR     SKIPSP
04050 035B BD 042A        JSR     TSTLTR
04060 035E 25 26          BCS     FINV11
04070 0360 16       FINV1 TAB
04080 0361 A6 01          LDA A   1,X
04090 0363 BD 042A        JSR     TSTLTR
04100 0366 25 20          BCS     FINV2
04110 0368 BD 0789        JSR     FCODE
04120 036B BD 02B2        JSR     PUSHX
04130 036E DE 34          LDX     BASPNT
04140 0370 BD 04B1        JSR     PAREXP
04150 0373 DF 34          STX     BASPNT
04160 0375 BD 02C7        JSR     PULLX
```

```
04170 0378 FE 00          LDX    0,X
04180 037A AD 00          JSR    0,X
04190 037C BD 043F        JSR    PUSHAF
04200 037F DF 34          LDX    BASPNT
04210 0381 31             INS                  SINCE FUNCTION CAN ONLY BE
04220 0382 31             INS                  ON RIGHT OF EXPR WE SKIP
04230 0383 31             INS
04240 0384 31             INS                  DIRECTLY FROM TERM
04250 0385 39             RTS
04260 0386 0B       FINV11 SEV
04270 0387 39             RTS
04280 0388 BD 0432 FINV2  JSR    TESTNO
04290 038B 24 07          BCC    FINV3
04300 038D 81 28          CMP A  #'(
04310 038F 27 03          BEQ    FINV3
04320 0391 09             DEX
04330 0392 86 20          LDA A  #$20
04340 0394 08       FINV3 INX
04350 0395 08             INX
04360 0396 BD 02B2        JSR    PUSHX
04370 0399 DE 2A          LDX    NEXTBA
04380 039B 9C 46    FINV4 CPX    ARRTAB
04390 039D 0D             SEC
04400 039E 27 09          BEQ    FINV5
04410 03A0 E1 00          CMP B  0,X
04420 03A2 26 0C          BNE    FINV6
04430 03A4 A1 01          CMP A  1,X
04440 03A6 26 08          BNE    FINV6
04450 03A8 0C             CLC
04460 03A9 DF 63    FINV5 STX    VARPNT
04470 03AB BD 02C7        JSR    PULLX
04480 03AE 0A             CLV
04490 03AF 39             RTS

04510 03B0 36       FINV6 PSH A
04520 03B1 A6 01          LDA A  1,X
04530 03B3 81 28          CMP A  #'(
04540 03B5 32             PUL A
04550 03B6 27 05          BEQ    FINV7
04560 03B8 BD 0457        JSR    INXATF
04570 03BB 20 DF          BRA    FINV4
04580 03BD EE 04    FINV7 LDX    4,X
04590 03BF 20 DA          BRA    FINV4



04610 03C1 BD 0281 TSTV   JSR    BREAK
04620 03C4 8D 92          BSR    FINVAR
04630 03C6 28 01          BVC    TSTV1
04640 03C8 39             RTS
04650 03C9 36       TSTV1 PSH A
04660 03CA 25 31          BCS    TSTV4
04670 03CC DF 34          STX    BASPNT
04680 03CE DF 63          LDX    VARPNT
04690 03D0 81 28          CMP A  #'(
04700 03D2 27 15          BEQ    TSTV3
04710 03D4 08       TSTV11 INX
04720 03D5 08             INX
04730 03D6 DF 63    TSTV2 STX    VARPNT
04740 03D8 DF 5D          LDX    AESTK
04750 03DA 96 63          LDA A  VARPNT
04760 03DC A7 00          STA A  0,X
04770 03DE 96 64          LDA A  VARPNT+1
04780 03E0 A7 01          STA A  1,X
04790 03E2 8D 5A          BSR    PUSHAF
04800 03E4 32             PUL A
04810 03E5 DF 34          LDX    BASPNT
```

```
04820 03E7 0C              CLC
04830 03E8 39              RTS
04840 03E9 BD 0DE4 TSTV3   JSR     GETPAR
04850 03EC 24 04           BCC     TSTV33
04860 03EE 8D 77    TSTV31 BSR     FIX
04870 03F0 20 01           BRA     TSTV34
04880 03F2 5F       TSTV33 CLR B
04890 03F3 37       TSTV34 PSH B
04900 03F4 8D 71           BSR     FIX
04910 03F6 17              TRA
04920 03F7 33              PUL B
04930 03F8 BD 0F59         JSR     SUBCAL
04940 03FB 20 D9           BRA     TSTV2
04950 03FD DF 34    TSTV4  STX     BASPNT
04960 03FF DE 63           LDX     VARPNT
04970 0401 E7 00           STA B   0,X
04980 0403 A7 01           STA A   1,X
04990 0405 81 28           CMP A   #'(
05000 0407 26 19           BNE     TSTV5
05010 0409 C6 0A           LDA B   #10
05020 040B E7 02           STA B   2,X
05030 040D E7 03           STA B   3,X
05040 040F BD 0DE4         JSR     GETPAR
05050 0412 25 03           BCS     TSTV44
05060 0414 5F              CLR B
05070 0415 E7 03           STA B   3,X
05080 0417 A6 02    TSTV44 LDA A   2,X
05090 0419 BD 0F41         JSR     DIMCAL
05100 041C 6D 03           TST     3,X
05110 041E 27 D2           BEQ     TSTV33
05120 0420 20 CC           BRA     TSTV31
05130 0422 8D 33    TSTV5  BSR     INXATF
05140 0424 DF 46           STX     ARRTAB
05150 0426 DE 63           LDX     VARPNT
05160 0428 20 AA           BRA     TSTV11
05190 042A 81 41    TSTLTR CMP A   #$41
05200 042C 2B 0C           BMI     NONO
05210 042E 81 5A           CMP A   #$5A
05220 0430 2F 0A           BLE     YESNO


05240 0432 81 30    TESTNO CMP A   #$30
05250 0434 2B 04           BMI     NONO
05260 0436 81 39           CMP A   #$39
05270 0438 2F 02           BLE     YESNO
05280 043A 0D       NONO   SEC
05290 043B 39              RTS
05300 043C 0C       YESNO  CLC
05310 043D 39              RTS




05330 043E DF 5D    PUSHAE LDX     AESTK
05340 0440 8D 16           BSR     INXSEV
05350 0442 DF 5D           STX     AESTK
05360 0444 39              RTS


05380 0445 BD 02B2 PSHXPA JSR     PUSHX

05400 0448 DF 5D    PULLAE LDX     AESTK
05410 044A 8D 03           BSR     DEXSEV
05420 044C 20 F4           BRA     *-10


05440 044E 09       DEXATE DEX
05450 044F 09       DEXSEV DEX
```

```
05460 0450 09       DEXSIX DEX
05470 0451 09       DEXFIV DEX
05480 0452 09       DEXFOR DEX
05490 0453 09              DEX
05500 0454 09              DEX
05510 0455 09              DEX
05520 0456 39              RTS


05540 0457 08       INXATE INX
05550 0458 08       INXSEV INX
05560 0459 08       INXSIX INX
05570 045A 08       INXFIV INX
05580 045B 08       INXFOR INX
05590 045C 08              INX
05600 045D 08              INX
05610 045E 08              INX
05620 045F 9C 44    OVRTST CPX    MEMEND
05630 0461 2B 03           BMI    *+5
05640 0463 7E 09FB         JMP    OVERFL
05650 0466 39              RTS




05670 0467 BD 0445 FIX     JSR    PSHXPA
05680 046A A6 06           LDA A  6,X
05690 046C 81 04           CMP A  #4
05700 046E 2F 05           BLE    *+7
05710 0470 C6 01    FIXERR LDA B  #1
05720 0472 7E 0828         JMP    ERROR
05730 0475 4F              CLR A
05740 0476 5F              CLR B
05750 0477 6D 06    FIX2   TST    6,X
05760 0479 2F 19           BLE    FIXEX
05770 047B 58              ASL B
05780 047C 49              ROL A
05790 047D D7 52           STA B  BNUMB
05800 047F 97 51           STA A  ANUMB
05810 0481 58              ASL B
05820 0482 49              ROL A
05830 0483 58              ASL B
05840 0484 49              ROL A
05850 0485 DB 52           ADD B  BNUMB
05860 0487 99 51           ADC A  ANUMB
05870 0489 EB 00           ADD B  0,X
05880 048B 89 00           ADC A  #0
05890 048D BD 0552         JSR    ALLEFT
05900 0490 6A 06           DEC    6,X
05910 0492 20 F3           BRA    FIX2
05920 0494 4D       FIXEX  TST A
05930 0495 26 D9           BNE    FIXERR
05940 0497 7E 02C7         JMP    PULLX



05960 049A BD 0744 FACT    JSR    SKIPSP
05970 049D BD 03C1         JSR    TSTV
05980 04A0 25 04           BCS    *+6
05990 04A2 BD 0309         JSR    IND
06000 04A5 39              RTS
06010 04A6 BD 0B28         JSR    TSTN
06020 04A9 25 01           BCS    *+3
06030 04AB 39              RTS
06040 04AC 81 28           CMP A  #'(
06050 04AE 26 11           BNE    FACT3
06060 04B0 08              INX
```

```
06070 04B1 8D 33    PAREXP BSR     EXPR
06080 04B3 BD 0744         JSR     SKIPSP
06090 04B6 81 29           CMP A   #')
06100 04B8 26 02           BNE     FACT2
06110 04BA 08              INX
06120 04BB 39              RTS
06130 04BC C6 13    FACT2  LDA B   #$13
06140 04BE 7F 0828         JMP     ERROR
06150 04C1 C6 06    FACT3  LDA B   #$06
06160 04C3 20 F9           BRA     *-5



06180 04C5 8D D3    TERM   BSR     FACT
06190 04C7 BD 0744  TERM0  JSR     SKIPSP
06200 04CA 81 2A           CMP A   #'*
06210 04CC 26 0B           BNE     TERM1
06220 04CE 08              INX
06230 04CF BD 0744         JSR     SKIPSP
06240 04D2 8D C6           BSR     FACT
06250 04D4 BD 06D1         JSR     MPY
06260 04D7 20 EE           BRA     TERM0
06270 04D9 81 2F    TERM1  CMP A   #'/
06280 04DB 26 08           BNE     TERM2
06290 04DD 08              INX
06300 04DE 8D BA           BSR     FACT
06310 04E0 BD 0659         JSR     DIV
06320 04E3 20 E2           BRA     TERM0
06330 04E5 39       TERM2  RTS



06350 04E6 BD 0744  EXPR   JSR     SKIPSP
06360 04E9 81 2D           CMP A   #'-
06370 04EB 26 07           BNE     *+9
06380 04ED 08              INX
06390 04EE 8D D5           BSR     TERM
06400 04F0 8D 2D           BSR     NEG
06410 04F2 20 07           BRA     EXPR1
06420 04F4 81 2B           CMP A   #'+
06430 04F6 26 01           BNE     *+3
06440 04F8 08              INX
06450 04F9 8D CA           BSR     TERM
06460 04FB BD 0744  EXPR1  JSR     SKIPSP
06470 04FE 81 2B           CMP A   #'+
06480 0500 26 08           BNE     *+10
06490 0502 08              INX
06500 0503 8D C0           BSR     TERM
06510 0505 BD 05EC         JSR     ADD
06520 0508 20 F1           BRA     EXPR1
06530 050A 81 2D           CMP A   #'-
06540 050C 26 08           BNE     *+10
06550 050E 08              INX
06560 050F 8D B4           BSR     TERM
06570 0511 BD 05F9         JSR     SUB
06580 0514 20 F5           BRA     EXPR1
06590 0516 39              RTS



06610 0517 BD 02B2  NEGOP  JSR     PUSHX
06620 051A BD 0458         JSR     INXSEV
06630 051D 20 0C           BRA     *+14
```

```
06640 051F BD 02B2 NEG     JSR     PUSHX
06650 0522 DF 5D           LDX     AESTK
06660 0524 09              DEX
06670 0525 09              DEX
06680 0526 20 06           BRA     **8
06690 0528 BD 02B2 NEGACC  JSR     PUSHX
06700 052B BD 045A         JSR     INXFIV
06710 052E 36              PSH A
06720 052F 37              PSH B
06730 0530 C6 06           LDA B   #6
06740 0532 86 99    NEGA11 LDA A   #$99
06750 0534 A0 00           SUB A   0,X
06760 0536 A7 00           STA A   0,X
06770 0538 09              DEX
06780 0539 5A              DEC B
06790 053A 26 F6           BNE     NEGA11
06800 053C BD 0459         JSR     INXSTX
06810 053F 0D              SEC
06820 0540 C6 06           LDA B   #6
06830 0542 86 00    NEGA2  LDA A   #$00
06840 0544 A9 00           ADC A   0,X
06850 0546 19              DAA
06860 0547 A7 00           STA A   0,X
06870 0549 09              DEX
06880 054A 5A              DEC B
06890 054B 26 F5           BNE     NEGA2
06900 054D 33              PUL B
06910 054E 32              PUL A
06920 054F 7E 02C7         JMP     PULLX



06940 0552 36       ALLEFT PSH A
06950 0553 8D 08           BSR     LLEFT
06960 0555 A6 00           LDA A   0,X
06970 0557 84 0F           AND A   #$0F
06980 0559 A7 00           STA A   0,X
06990 055B 32              PUL A
07000 055C 39              RTS

07020 055D 36       LLEFT  PSH A
07030 055E 37              PSH B
07040 055F DF 57           STX     XLONG
07050 0561 C6 04           LDA B   #4
07060 0563 DF 57    LLEFT2 LDX     XLONG
07070 0565 BD 045A         JSR     INXFIV
07080 0568 68 00           ASL     0,X
07090 056A 86 05           LDA A   #5
07100 056C 09       LLEFT3 DEX
07110 056D 69 00           ROL     0,X
07120 056F 4A              DEC A
07130 0570 26 FA           BNE     LLEFT3
07140 0572 5A              DEC B
07150 0573 26 EE           BNE     LLEFT2
07160 0575 33              PUL B
07170 0576 32              PUL A
07180 0577 39              RTS



07200 0578 36       LRIGHT PSH A
07210 0579 37              PSH B
07220 057A DF 57           STX     XLONG
07230 057C A6 00           LDA A   0,X
```

```
07240 057E 84 F0          AND A  #$F0
07250 0580 97 53          STA A  LRNINF
07260 0582 C6 04          LDA B  #4
07270 0584 DE 57    LRIGH2 LDX    XLONG
07280 0586 64 00          LSR    0,X
07290 0588 86 05          LDA A  #5
07300 058A 08       LRIGH3 INX
07310 058B 66 00          ROR    0,X
07320 058D 4A             DEC A
07330 058E 26 FA          BNE    LRIGH3
07340 0590 5A             DEC B
07350 0591 26 F1          BNE    LRIGH2
07360 0593 DE 57          LDX    XLONG
07370 0595 A6 00          LDA A  0,X
07380 0597 9B 53          ADD A  LRNINF
07390 0599 A7 00          STA A  0,X
07400 059B 33             PUL B
07410 059C 32             PUL A
07420 059D 39             RTS



07440 059E BD 043F NORMPA JSR    PUSHAF

07460 05A1 BD 0445 NORM   JSR    PSHXPA
07470 05A4 7F 0056        CLR    NORSGN
07480 05A7 BD 0DA4        JSR    TSTZER
07490 05AA 24 19          BCC    NORMEZ
07500 05AC A6 00          LDA A  0,X
07510 05AE 2A 0B          BPL    NORM1
07520 05B0 97 56          STA A  NORSGN
07530 05B2 BD 0528        JSR    NEGACC
07540 05B5 86 0F    NORM00 LDA A  #$0F
07550 05B7 A4 00          AND A  0,X
07560 05B9 A7 00          STA A  0,X
07570 05BB A6 00    NORM1  LDA A  0,X
07580 05BD 26 08          BNE    NORMEX
07590 05BF 8D 91          BSR    ALLEFT
07600 05C1 6A 06          DEC    6,X
07610 05C3 20 F6          BRA    NORM1
07620 05C5 6F 06    NORMEZ CLR    6,X
07630 05C7 6F 05    NORMEX CLR    5,X
07640 05C9 A6 06          LDA A  6,X
07650 05CB 81 65          CMP A  #101
07660 05CD 2D 15          BLT    NORMF0
07670 05CF 37             PSH B
07680 05D0 C6 05          LDA B  #5
07690 05D2 86 99          LDA A  #$99
07700 05D4 A7 00    NORM01 STA A  0,X
07710 05D6 08             INX
07720 05D7 5A             DEC B
07730 05D8 26 FA          BNE    NORM01
07740 05DA 86 64          LDA A  #100
07750 05DC A7 01          STA A  1,X
07760 05DE BD 0451        JSR    DEXFIV
07770 05E1 33             PUL B
07780 05E2 20 D1          BRA    NORM00
07790 05E4 81 9D    NORMF0 CMP A  #-99
07800 05E6 2F 03          BGT    NORMF1
07810 05E8 BD 0BDD        JSR    CLRACC
07820 05EB 7D 0056 NORMF1 TST    NORSGN
07830 05EE 27 03          BEQ    NORMF2
07840 05F0 BD 0528        JSR    NEGACC
07850 05F3 BD 043F NORMF2 JSR    PUSHAF
07860 05F6 7F 02C7        JMP    PULLX

07880 05F9 BD 051F SUB    JSR    NEG
```

```
07900 05FC BD 0445 ADD     JSR     PSHXPA
07910 05FF BD 0DA4         JSR     TSTZFR
07920 0602 24 25           BCC     ADD4
07930 0604 BD 0578         JSR     LRIGHT
07940 0607 6C 06           INC     6,X
07950 0609 BD 0448         JSR     PULLAF
07960 060C BD 0DA4         JSR     TSTZFR
07970 060F 24 11           BCC     ADD1
07980 0611 BD 0578 ADD0    JSR     LRIGHT
07990 0614 6C 06           INC     6,X
08000 0616 A6 06           LDA A   6,X
08010 0618 A1 0D           CMP A   13,X
08020 061A 27 08           BEQ     ADD2
08030 061C 2D F3           BLT     ADD0
08040 061E 8D 0C           BSR     SWAP
08050 0620 20 EF           BRA     ADD0
08060 0622 8D 08   ADD1    BSR     SWAP
08070 0624 8D 1F   ADD2    BSR     ADDER
08080 0626 BD 059F         JSR     NORMPA
08090 0629 7E 02C7 ADD4    JMP     PULLX



08110 062C DE 5D   SWAP    LDX     AFSTK
08120 062E 36              PSH A
08130 062F 37              PSH B
08140 0630 C6 07           LDA B   #7
08150 0632 A6 00   SWAP1   LDA A   0,X
08160 0634 36              PSH A
08170 0635 A6 07           LDA A   7,X
08180 0637 A7 00           STA A   0,X
08190 0639 32              PUL A
08200 063A A7 07           STA A   7,X
08210 063C 08              INX
08220 063D 5A              DEC B
08230 063E 26 F2           BNE     SWAP1
08240 0640 DE 5D   SWAP2   LDX     AFSTK
08250 0642 33              PUL B
08260 0643 32              PUL A
08270 0644 39              RTS



08290 0645 36      ADDER   PSH A
08300 0646 37              PSH B
08310 0647 DE 5D           LDX     AFSTK
08320 0649 C6 06           LDA B   #6
08330 064B 0C              CLC
08340 064C A6 05   ADDER1  LDA A   5,X
08350 064E A9 0C           ADC A   12,X
08360 0650 19              DAA
08370 0651 A7 05           STA A   5,X
08380 0653 09              DEX
08390 0654 5A              DEC B
08400 0655 26 F5           BNE     ADDER1
08410 0657 20 E7           BRA     SWAP2



08430 0659 BD 02B2 DIV     JSR     PUSHX
08440 065C BD 0BDD         JSR     CLRACC
08450 065F BD 0448         JSR     PULLAF
08460 0662 BD 0DA4         JSR     TSTZFR
```

```
08470 0665 25 05          BCS    *+7
08480 0667 C6 08          LDA B  #$08
08490 0669 7E 0828        JMP    ERROR
08500 066C 60 06          NEG    6,X
08510 066E 8D 38          BSR    MDSIGN
08520 0670 BD 0578        JSR    LRIGHT
08530 0673 BD 0517        JSR    NEGOP
08540 0676 86 0B          LDA A  #11
08550 0678 5F       DIV2  CLR B
08560 0679 5C       DIV3  INC B
08570 067A 8D C9          BSR    ADDER
08580 067C 6D 00          TST    0,X
08590 067E 2A F9          BPL    DIV3
08600 0680 5A             DEC B
08610 0681 BD 0517        JSR    NEGOP
08620 0684 8D BF          BSR    ADDER
08630 0686 BD 0517        JSR    NEGOP
08640 0689 BD 0458        JSR    INXSEV
08650 068C BD 0458        JSR    INXSEV
08660 068F BD 055D        JSR    LLEFT
08670 0692 FB 05          ADD B  5,X
08680 0694 E7 05          STA B  5,X
08690 0696 DE 5D          LDX    AESTK
08700 0698 BD 055D        JSR    LLEFT
08710 069B 4A             DEC A
08720 069C 26 DA          BNE    DIV2
08730 069E BD 043E        JSR    PUSHAE
08740 06A1 8D 89          BSR    SWAP
08750 06A3 7C 0050        INC    TNUMB
08760 06A6 20 56          BRA    MDEXIT
08780 06A8 BD 0578 MDSIGN JSR    LRIGHT
08790 06AB 4F             CLR A
08800 06AC 6D 00          TST    0,X
08810 06AE 2A 05          BPL    MDS2
08820 06B0 BD 0528        JSR    NEGACC
08830 06B3 86 80          LDA A  #$80
08840 06B5 BD 0448 MDS2   JSR    PULLAE
08850 06B8 6D 00          TST    0,X
08860 06BA 2A 05          BPL    MDS3
08870 06BC BD 0528        JSR    NEGACC
08880 06BF 8B 80          ADD A  #$80
08890 06C1 97 4A    MDS3  STA A  TSIGN
08900 06C3 A6 06          LDA A  6,X
08910 06C5 AB 0D          ADD A  13,X
08920 06C7 28 05          BVC    MDS4
08930 06C9 86 78          LDA A  #120
08940 06CB 24 01          BCC    MDS4
08950 06CD 40             NEG A
08960 06CE 97 50    MDS4  STA A  TNUMB
08970 06D0 39             RTS


08990 06D1 BD 0445 MPY    JSR    PSHXPA
09000 06D4 8D D2          BSR    MDSIGN
09010 06D6 DF 59          STX    MPYXX
09020 06D8 BD 043E        JSR    PUSHAE
09030 06DB BD 062C        JSR    SWAP
09040 06DE BD 0RDD        JSR    CLRACC
09050 06E1 86 09          LDA A  #9
09060 06E3 DE 59    MPY4  LDX    MPYXX
09070 06E5 E6 00          LDA B  0,X
09080 06E7 BD 0552        JSR    ALLEFT
09090 06EA 5D       MPY5  TST B
09100 06EB 27 06          BEQ    MPY6
09110 06ED BD 0645        JSR    ADDER
09120 06F0 5A             DEC B
```

```
09130 06F1 20 F7          BRA     MPY5
09140 06F3 DF 5D    MPY6  LDX     AESTK
09150 06F5 BD 0458        JSR     INXSEV
09160 06F8 BD 0578        JSR     LRIGHT
09170 06FB 4A             DEC A
09180 06FC 26 F5          BNE     MPY4
09190 06FE BD 0448  MDEXIT JSR    PULLAE
09200 0701 BD 062C        JSR     SWAP
09210 0704 96 50          LDA A   TNUMR
09220 0706 A7 06          STA A   6,X
09230 0708 BD 059F        JSR     NORMPA
09240 070B 7D 004A        TST     TSIGN
09250 070E 2A 03          BPL     MDEX2
09260 0710 BD 051F        JSR     NEG
09270 0713 7E 02C7  MDEX2 JMP     PULLX


09300 0716 96 32    FINDNO LDA A  HIGHLN
09310 0718 D6 33           LDA B  HIGHLN+
09320 071A D0 31           SUB B  PACKLN+
09330 071C 92 30           SBC A  PACKLN
09340 071E 25 1E           BCS    HIBALL
09350 0720 DF 2F    FINDN1 LDX    SOURCE
09360 0722 96 30    FINDO  LDA A  PACKLN
09370 0724 D6 31           LDA B  PACKLN+
09380 0726 F0 01           SUB B  1,X
09390 0728 A2 00           SBC A  0,X
09400 072A 25 14           BCS    *+22
09410 072C 26 03           BNE    *+5
09420 072E 5D              TST B
09430 072F 27 10           BEQ    *+18
09440 0731 08              INX
09450 0732 08              INX
09460 0733 A6 00           LDA A  0,X
09470 0735 81 1F           CMP A  #$1F
09480 0737 26 F9           BNE    *-5
09490 0739 08              INX
09500 073A 9C 2A           CPX    NEXTBA
09510 073C 26 E4           BNE    FINDO
09520 073E DF 2A    HIBALL LDX    NEXTRA
09530 0740 0D              SEC
09540 0741 DF 2C           STX    WORKRA
09550 0743 39              RTS


09570 0744 A6 00    SKIPSP LDA A  0,X
09580 0746 81 20           CMP A  #$20
09590 0748 26 03           BNE    *+5
09600 074A 08       INXSKP INX
09610 074B 20 F7           BRA    SKIPSP
09620 074D 39              RTS


09640 074E BD 0B28 LINENO JSR     TSTN
09650 0751 24 05          BCC     *+7
09660 0753 C6 07    LINEO LDA B   #7
09670 0755 7E 0828        JMP     ERROR
09680 0758 DF AC          STX     BUFNXT
09690 075A BD 0448        JSR     PULLAE
09700 075D E6 06          LDA B   6,X
09710 075F 23 F2          BLS     LINEO
09720 0761 C1 04          CMP B   #4
09730 0763 2E EE          BGT     LINEO
09740 0765 86 05          LDA A   #5
09750 0767 10             SBA
09760 0768 4D             TST A
09770 0769 27 06          BEQ     *+8
09780 076B BD 0578        JSR     LRIGHT
09790 076E 4A             DEC A
```

```
09800 076F 20 F7          BRA     *-7
09810 0771 A6 01          LDA A   1,X
09820 0773 97 30          STA A   PACKLN
09830 0775 A6 02          LDA A   2,X
09840 0777 97 31          STA A   PACKLN+1
09850 0779 DE AC          LDX     BUFNXT
09860 077B 0C             CLC
09870 077C 39             RTS



09890 077D DF 34   NXTLIN LDX     BASPNT
09900 077F A6 00          LDA A   0,X
09910 0781 08             INX
09920 0782 81 1F          CMP A   #$1F
09930 0784 26 F9          BNE     *-5
09940 0786 DF 36          STX     BASLIN
09950 0788 39             RTS

09970 0789 DF AC   FCODE  STX     BUFNXT
09980 078B 9F 28          STS     SAVESP
09990 078D CE 01BC        LDX     #FUNCT-1
10000 0790 20 12          BRA     LOOP3

10020 0792 DF AC   CCINT  STX     BUFNXT
10030 0794 9F 28          STS     SAVESP
10040 0796 CE 0105        LDX     #INTER-1
10050 0799 20 09          BRA     LOOP3


10070 079B 8D A7   CCODE  BSR     SKIPSP
10080 079D DF AC          STX     BUFNXT
10090 079F 9F 28          STS     SAVESP
10100 07A1 CE 0111        LDX     #COMMAN-1
10110 07A4 9F AC   LOOP3  LDS     BUFNXT
10120 07A6 34             DES
10130 07A7 08             INX
10140 07A8 32             PUL A
10150 07A9 81 20          CMP A   #$20
10160 07AB 27 FB          BEQ     *-3
10170 07AD E6 00          LDA B   0,X
10180 07AF C1 1F          CMP B   #$1F
10190 07B1 27 18          BEQ     *+26
10200 07B3 11             CBA
10210 07B4 27 F1          BEQ     *-13
10220 07B6 08      LOOP5  INX
10230 07B7 8C 01BA        CPX     #COMEND

10240 07BA 27 17          BEQ     CCEXIT
10250 07BC 8C 01ED        CPX     #FUNEND
10260 07BF 27 18          BEQ     FUNCEX
10270 07C1 E6 00          LDA B   0,X
10280 07C3 C1 1F          CMP B   #$1F
10290 07C5 26 EF          BNE     LOOP5
10300 07C7 08             INX
10310 07C8 08             INX
10320 07C9 20 D9          BRA     LOOP3
10330 07CB 08             INX
10340 07CC 9F AC          STS     BUFNXT
10350 07CE 9F 34          STS     BASPNT
10360 07D0 9F 28          LDS     SAVESP
10370 07D2 39             RTS
10380 07D3 9F 28   CCEXIT LDS     SAVESP
```

```
10390 07D5 CE 01BB          LDX    #IMPLET
10400 07D8 39               RTS
10410 07D9 7E 0B18 FUNCFX   JMP    DRLLTR
10440 07DC CE 1200 NEWSUB   LDX    #FILE
10450 07DF DF 2E            STX    SOURCE
10460 07E1 DF 2A            STX    NEXTBA
10470 07E3 DF 46            STX    ARRTAB
10480 07E5 4F               CLR A
10490 07E6 97 32            STA A  HIGHLN
10500 07E8 97 33            STA A  HIGHLN+1
10510 07EA 97 40            STA A  PRCNT
10520 07EC CE 1078          LDX    #DUMRTS
10530 07EF DF 67            STX    DUMMY
10540 07F1 39               RTS



10560                       *START OF MAIN PROGRAM
10570                       *
10580                       *
10590 07F2 8D E8   START     BSR    NEWSUB
10600 07F4 20 65            BRA    RUN
10610 07F6 8E A045 READY    LDS    #$A045
10620 07F9 CE A07F          LDX    #$A07F
10630 07FC DF 3A            STX    XSTACK
10640 07FE 7F 0085          CLR    LOAFLG
10650 0801 CE 01F0          LDX    #RDYMSG
10660 0804 BD 0292          JSR    OUTPUT
10670 0807 7F 0040 NEWLIN   CLR    PRCNT
10680 080A BD 025D          JSR    CNTLIN
10690 080D CE 00B0          LDX    #BUFFER
10700 0810 BD 0744          JSR    SKIPSP
10710 0813 BD 0432          JSR    TESTNO
10720 0816 25 05            BCS    *+7
10730 0818 BD 0949          JSR    NUMBER
10740 081B 20 EA            BRA    NEWLIN
10750 081D 81 1F            CMP A  #$1F
10760 081F 27 E6            BEQ    NEWLIN
10770 0821 BD 079B          JSR    CODDE
10780 0824 FE 00            LDX    0,X
10790 0826 6E 00            JMP    0,X



10810 0828 8E A045 ERROR    LDS    #$A045
10820 082B BD 02A0          JSR    CRLF
10830 082E CE 020C          LDX    #ERRMS1
10840 0831 BD 0299          JSR    OUTNCR
10850 0834 D7 20            STA B  INDEX1
10860 0836 CE 0020          LDX    #INDEX1
10870 0839 BD 0270          JSR    OUT2H
10880 083C CE 0214          LDX    #ERRMS2
10890 083F BD 0299          JSR    OUTNCR
10900 0842 DF 5B            LDX    BASNUM
10910 0844 FE 00            LDX    0,X
10920 0846 DF 30            STX    PACKLN
10930 0848 CE 0030          LDX    #PACKLN
10940 084B 96 34            LDA A  BASPNT
10950 084D 26 04            BNE    *+6
10960 084F 6F 00            CLR    0,X
10970 0851 6F 01            CLR    1,X
10980 0853 BD 0273 ERROR2   JSR    OUT4HS
10990 0856 BD 02A0          JSR    CRLF
11000 0859 20 9B            BRA    READY
```

```
11020 085B DF 2F    RUN     LDX     SOURCE   WITH TEXT BUFFER POINTER
11030 085D DF 3C            STX     DATPNT
11040 085F DF 36            STX     BASLIN
11050 0861 CE 0075          LDX     #SRSTK
11060 0864 DF 65            STX     SRPNT
11070 0866 CE 1180          LDX     #FORSTK
11080 0869 DF 5F            STX     FORPNT
11090 086B DE 2A            LDX     NEXTBA
11100 086D DF 46            STX     ARRTAB
11110 086F 4F               CLR A
11120 0870 09               DEX
11130 0871 08               INX
11140 0872 A7 00            STA A   0,X
11150 0874 9C 44            CPX     MEMEND
11160 0876 27 07            BEQ     *+9
11170 0878 A6 00            LDA A   0,X
11180 087A 27 F5            BEQ     *-9
11190 087C 09               DEX
11200 087D DF 44            STX     MEMEND
11210 087F 7E 0A00          JMP     BASIC



11230 0882 CE 0200 CLIST    LDX     #PGCNTL
11240 0885 BD 0292          JSR     OUTPUT
11250 0888 DE 34            LDX     BASPNT
11260 088A BD 0744          JSR     SKIPSP
11270 088D 81 1F            CMP A   #$1F
11280 088F 27 2B            BEQ     CLIST4
11290 0891 BD 074F          JSR     LINENO
11300 0894 DF 34            STX     BASPNT
11310 0896 BD 0720          JSR     FINDNI
11320 0899 DF 24            STX     INDEX3
11330 089B DE 34            LDX     BASPNT
11340 089D BD 0744          JSR     SKIPSP
11350 08A0 81 1F            CMP A   #$1F
11360 08A2 27 04            BEQ     CLIST3
11370 08A4 08               INX
11380 08A5 BD 074F          JSR     LINENO
11390 08A8 4F       CLIST3  CLR A
11400 08A9 C6 01            LDA B   #1
11410 08AB DB 31            ADD B   PACKLN+
11420 08AD 19               DAA
11430 08AE D7 31            STA B   PACKLN+
11440 08B0 99 30            ADC A   PACKLN
11450 08B2 19               DAA
11460 08B3 97 30            STA A   PACKLN
11470 08B5 BD 0720          JSR     FINDNI
11480 08B8 DE 24            LDX     INDEX3
11490 08BA 20 06            BRA     CLIST5
11500 08BC DE 2A    CLIST4  LDX     NEXTBA
11510 08BE DF 2C            STX     WORKBA
11520 08C0 DE 2F            LDX     SOURCE
11530 08C2 9C 2C    CLIST5  CPX     WORKBA
11540 08C4 27 0A            BEQ     CLEXT
11550 08C6 9C 2A            CPX     NEXTBA
11560 08C8 27 06            BEQ     CLEXT
11570 08CA BD 032A          JSR     OUTLIN
11580 08CD 08               INX
11590 08CE 20 F2            BRA     CLIST5
11600 08D0 7E 0DDF CLEXT    JMP     REMARK


11620 08D3 BD 077D PATCH    JSR     NXTLIN
11630 08D6 CE 0A00          LDX     #BASIC
11640 08D9 FF A046          STX     $A046
```

```
11650 08DC 8E A040          LDS     #$A040
11660 08DF BF A008          STS     $A008
11670 08E2 7E F0E3          JMP     $F0E3



11690 08E5 DE 2F    SAVE    LDX     SOURCE
11700 08E7 86 12            LDA A   #$12
11710 08E9 BD 0276          JSR     OUTCH
11720 08EC 8D 45            BSR     DELAY2
11730 08EE 8D 43            BSR     DELAY2
11740 08F0 8D 4C            BSR     LEADER
11750 08F2 9C 2A    SAVE1   CPX     NEXTRA
11760 08F4 27 0D            BEQ     SAVE5
11770 08F6 86 02            LDA A   #$02
11780 08F8 BD 0276          JSR     OUTCH
11790 08FB BD 032A          JSR     OUTLIN
11800 08FE 08               INX
11810 08FF 8D 34            BSR     DELAY
11820 0901 20 FF            BRA     SAVE1
11830 0903 86 03    SAVE5   LDA A   #$03
11840 0905 BD 0276          JSR     OUTCH
11850 0908 8D 34            BSR     LEADER
11860 090A 7E 07F6          JMP     READY



11880 090D BD 07DC LOAD     JSR     NEWSUB
11890 0910 86 11    APPEND  LDA A   #$11
11900 0912 97 85            STA A   LOADFLG
11910 0914 BD 0276          JSR     OUTCH
11920 0917 BD 027B LOAD1    JSR     INCH
11930 091A 81 02            CMP A   #$02
11940 091C 26 F9            BNE     LOAD1
11950 091E BD 027B          JSR     INCH
11960 0921 BD 0432          JSR     TESTNO
11970 0924 25 F4            BCS     LOAD1+3
11980 0926 CE 00B0          LDX     #BUFFER
11990 0929 BD 0231          JSR     KEYBD0+5
12000 092C CE 00B0          LDX     #BUFFER
12010 092F 8D 18            BSR     NUMBER
12020 0931 20 F4            BRA     LOAD1

12040 0933 8D 00    DELAY2  BSR     DELAY

12060 0935 5F       DELAY   CLR B
12070 0936 4F               CLR A
12080 0937 4A       DELAY3  DEC A
12090 0938 26 FD            BNE     DELAY3
12100 093A 5A               DEC B
12110 093B 26 F9            BNE     DELAY+1
12120 093D 39               RTS



12140 093E C6 32    LEADER  LDA B   #50
12150 0940 86 FF            LDA A   #$FF
12160 0942 BD 0276          JSR     OUTCH
12170 0945 5A               DEC B
12180 0946 26 F8            BNE     LEADER+2
12190 0948 39               RTS
```

```
12210 0949 BD 074F NUMBER JSR    LINENO
12220 094C BD 0716 NUM1   JSR    FINDNO
12230 094F 24 13          BCC    DELREP
12240       * YOU HAVE IDENTICAL NUMBER AND U MUST DELETE AND IN
12250 0951 DF AC          LDX    BUFNXT
12260 0953 BD 0744        JSR    SKIPSP
12270 0956 81 1F          CMP A  #$1F
12280 0958 27 21          BEQ    NEXIT
12290 095A DF 2C          LDX    WORKBA
12300 095C 9C 2A          CPX    NEXTBA
12310 095E 27 1C          BEQ    CAPPEN   IF AT END FILE
12320              * YOU MERELY APPEND
12330              * OTHERWISE JUST INSERT AND RETURN
12340 0960 8D 48          BSR    INSERT
12350 0962 20 17          BRA    NEXIT
12360 0964 DE AC   DELREP LDX    BUFNXT   GET FIRST CHARACTER
12370 0966 BD 0744        JSR    SKIPSP   AFTER LIN
12380 0969 81 1F          CMP A  #$1F     IS FIRST CHARACTER RS
12390 096B 26 0A          BNE    REPLAC   IF NOT YOU ARE
12400 096D DE 2A          LDX    NEXTBA   REPLACING LINE
12410 096F 9C 2E          CPX    SOURCE
12420 0971 27 08          BEQ    NEXIT
12430 0973 8D 0F          BSR    DELETE   OTHERWISE ONLY DELETE
12440 0975 20 04          BRA    NEXIT    AND RET
12450 0977 8D 0B   REPLAC BSR    DELETE
12460 0979 8D 2F          BSR    INSERT
12470 097B 39      NEXIT  RTS
12480 097C 8D 2C   CAPPEN BSR    INSERT
12490 097E DE 30          LDX    PACKLN   AND MAKE THIS LINE
12500 0980 DF 32          STX    HIGHLN   NEW HIGH L
12510 0982 20 F7          BRA    NEXIT


12530 0984 9F 28   DELETE STS    SAVESP
12540 0986 DE 2C          LDX    WORKBA
12550 0988 9E 2A          LDS    NEXTBA
12560 098A 08             INX
12570 098B 08             INX
12580 098C 34             DES
12590 098D 34             DES
12600 098E A6 00   DEL2   LDA A  0,X
12610 0990 34             DES
12620 0991 08             INX
12630 0992 81 1F          CMP A  #$1F     IF NOT RS,DO IT AGAIN
12640 0994 26 F8          BNE    DEL2
12650 0996 9F 2A          STS    NEXTBA   STACK WAS DECREMENTED
12660 0998 9F 46          STS    ARRTAB   BY # OF
12670 099A 35             TXS
12680 099B DE 2C          LDX    WORKBA
12690 099D 9C 2A   DEL4   CPX    NEXTBA
12700 099F 27 06          BEQ    DELEX
12710 09A1 32             PUL A
12720 09A2 A7 00          STA A  0,X      HONEST O OFFSET
12730 09A4 08             INX
12740 09A5 20 F6          BRA    DEL4
12750 09A7 9E 28   DELEX  LDS    SAVESP
12760 09A9 39             RTS



12780 09AA DE AC   INSERT LDX    BUFNXT
12790 09AC BD 079B        JSR    CCODE
12800 09AF 9F 28          STS    SAVESP
12810 09B1 DF 48          STX    KEYWD
12820 09B3 DE 2A          LDX    NEXTBA
12830 09B5 DF 20          STX    INDEX1
12840 09B7 D6 AF          LDA B  ENDBUF+1
```

```
12850 09B9 D0 AD          SUB B  BUFNXT+1
12860 09BB CB 04          ADD B  #$04
12870 09BD DB 2B          ADD B  NEXTRA+1
12880 09BF 86 00          LDA A  #$00
12890 09C1 99 2A          ADC A  NEXTBA
12900 09C3 91 44          CMP A  MEMEND
12910 09C5 22 34          BHI    OVERFL
12920 09C7 D7 2B          STA B  NEXTRA+1
12930 09C9 97 2A          STA A  NEXTBA    THIS INCREMENTS NEXTBA
12940 09CB DE 2A          LDX    NEXTBA    FOR OF
12950 09CD DF 46          STX    ARRTAB
12960 09CF 35             TXS
12970 09D0 DE 20          LDX    INDEX1
12980 09D2 9C 2C   INS2   CPX    WORKBA
12990 09D4 27 06          BEQ    BUFWRT
13000 09D6 09             DEX
13010 09D7 A6 00          LDA A  0,X
13020 09D9 36             PSH A
13030 09DA 20 F6          BRA    INS2
13040 09DC DE 2C   BUFWRT LDX    WORKBA    WE HAVE OPENED HOLE,NOW
13050                     *THE KEYBD BUFFER  CAN I
13060 09DE 96 30          LDA A  PACKLN
13070 09E0 A7 00          STA A  0,X
13080 09E2 08             INX
13090 09E3 96 31          LDA A  PACKLN+1
13100 09E5 A7 00          STA A  0,X
13110 09E7 08             INX
13120 09E8 96 49          LDA A  KEYWD+1
13130 09EA A7 00          STA A  0,X
13140 09EC 08             INX
13150 09ED 9F AC          LDS    BUFNXT
13160 09EF 34             DES
13170 09F0 32             PUL A
13180 09F1 A7 00          STA A  0,X
13190 09F3 08             INX
13200 09F4 81 1F          CMP A  #$1F
13210 09F6 26 F8          BNE    *-6
13220 09F8 9F 28          LDS    SAVESP
13230 09FA 39             RTS
13240 09FB C6 14   OVERFL LDA B  #$14
13250 09FD 7E 0828        JMP    ERROR
```

```
13270 0A00 CE 0112 BASIC  LDX    #COMMAN
13280 0A03 DF 48          STX    KEYWD
13290 0A05 CE 1102        LDX    #ASTACK
13300 0A08 DF 5D          STX    ASTK
13310 0A0A DF 36          LDX    BASLIN
13320 0A0C DF 5B          STX    BASNUM
13330 0A0E 9C 2A          CPX    NEXTBA
13340 0A10 26 03          BNE    *+5
13350 0A12 7E 07F6        JMP    READY
13360 0A15 7D 0036        TST    BASLIN
13370 0A18 27 F8          BEQ    *-6
13380 0A1A 08             INX
13390 0A1B 08             INX
13400 0A1C A6 00          LDA A  0,X
13410 0A1E 08             INX
13420 0A1F DF 34          STX    BASPNT
13430 0A21 97 49          STA A  KEYWD+1
13440 0A23 DE 48          LDX    KEYWD
13450 0A25 EE 00          LDX    0,X
13460 0A27 6E 00          JMP    0,X
```

```
13480 0A29 DE 34    THEN    LDX     BASPNT
13490 0A2B BD 0744          JSR     SKIPSP
13500 0A2E BD 0432          JSR     TESTNO
13510 0A31 24 49            BCC     GOTO
13520 0A33 7E 0FF2          JMP     IF2



13540 0A36 C6 20    ONERR   LDA B   #$20
13550 0A38 7E 0828          JMP     ERROR

13570 0A3B DE 34    ONGO    LDX     BASPNT
13580 0A3D BD 04E6          JSR     EXPR
13590 0A40 BD 0467          JSR     FIX
13600 0A43 5D               TST B
13610 0A44 27 F0            BEQ     ONERR
13620 0A46 5A               DEC B
13630 0A47 D7 55            STA B   ONLOOP
13640 0A49 BD 079B          JSR     CODE
13650 0A4C EE 00            LDX     0,X
13660 0A4E 8C 0A7C          CPX     #GOTO
13670 0A51 27 2C            BEQ     GOTO+3
13680 0A53 8C 0A5A          CPX     #GOSUB
13690 0A56 27 05            BEQ     GOSUB+3
13700 0A58 20 DC            BRA     ONERR



13720 0A5A 7F 0055  GOSUB   CLR     ONLOOP
13730 0A5D DE 36            LDX     BASLIN
13740 0A5F BD 077D          JSR     NXTLIN
13750 0A62 DF 65            LDX     SRRPNT
13760 0A64 8C 0085          CPX     #SRRSTK+16
13770 0A67 26 05            BNE     GOSUB1
13780 0A69 C6 09            LDA B   #9
13790 0A6B 7E 0828          JMP     ERROR
13800 0A6E 96 36    GOSUB1  LDA A   BASLIN
13810 0A70 A7 00            STA A   0,X
13820 0A72 08               INX
13830 0A73 96 37            LDA A   BASLIN+1
13840 0A75 A7 00            STA A   0,X
13850 0A77 08               INX
13860 0A78 DF 65            STX     SRRPNT
13870 0A7A 20 03            BRA     GOTO+3


13890 0A7C 7F 0055  GOTO    CLR     ONLOOP
13900 0A7F DE 34            LDX     BASPNT
13910 0A81 BD 074F  GOTO1   JSR     LINENO
13920 0A84 7A 0055          DEC     ONLOOP
13930 0A87 2B 0A            BMI     *+12
13940 0A89 BD 0744          JSR     SKIPSP
13950 0A8C 81 2C            CMP A   #',
13960 0A8E 26 A6            BNE     ONERR
13970 0A90 08               INX
13980 0A91 20 EF            BRA     GOTO1
13990 0A93 BD 0716          JSR     FINDNO
14000 0A96 24 05            BCC     GOTO2
14010 0A98 C6 07            LDA B   #7
14020 0A9A 7E 0828          JMP     ERROR
14030 0A9D DF 36    GOTO2   STX     BASLIN
14040 0A9F 7E 0A00          JMP     BASIC
```

```
14060 0AA2 DE 65    RETURN LDX    SBRPNT
14070 0AA4 8C 0075         CPX    #SBRSTK
14080 0AA7 26 05           BNE    **7
14090 0AA9 C6 10           LDA B  #$10
14100 0AAB 7E 0828         JMP    ERROR
14110 0AAE 09              DEX
14120 0AAF 09              DEX
14130 0AB0 DF 65           STX    SBRPNT
14140 0AB2 EE 00           LDX    0,X
14150 0AB4 20 E7           BRA    GOTO2




14180 0AB6 CE 01B3 STOP    LDX    #STOMSG
14190 0AB9 BD 0299         JSR    OUTNCR
14200 0ABC BD 0350         JSR    PRINSP
14210 0ABF DE 36           LDX    BASLIN
14220 0AC1 7E 0853         JMP    ERROR2




14240 0AC4 96 34    INPUT  LDA A  BASPNT
14250 0AC6 26 05           BNE    **7
14260 0AC8 C6 02           LDA B  #2
14270 0ACA 7E 0828 INPERR  JMP    ERROR
14280 0ACD BD 0225         JSR    KEYBD
14290 0AD0 CE 00B0         LDX    #BUFFER
14300 0AD3 DF AC           STX    BUFNXT
14310 0AD5 DE 34           LDX    BASPNT
14320 0AD7 BD 03C1 INPUT1  JSR    TSTV
14330 0ADA 25 31           BCS    INPEX
14340 0ADC DF 34           STX    BASPNT
14350 0ADE DE AC           LDX    BUFNXT
14360 0AE0 8D 3B    INPUT2 BSR    INNUM
14370 0AE2 24 12           BCC    INPUT4
14380 0AE4 81 1E           CMP A  #$1E
14390 0AE6 27 06           BEQ    **8
14400 0AE8 CE 0203         LDX    #WHAT
14410 0AEB BD 0292         JSR    OUTPUT
14420 0AEE BD 0225         JSR    KEYBD
14430 0AF1 CE 00B0         LDX    #BUFFER
14440 0AF4 20 EA           BRA    INPUT2
14450 0AF6 BD 02E0 INPUT4  JSR    STORE
14460 0AF9 BD 0744         JSR    SKIPSP
14470 0AFC 81 2C           CMP A  #',
14480 0AFE 26 01           BNE    INPUT5
14490 0B00 08              INX
14500 0B01 DF AC    INPUT5 STX    BUFNXT
14510 0B03 DE 34           LDX    BASPNT
14520 0B05 BD 0744         JSR    SKIPSP
14530 0B08 08              INX
14540 0B09 81 2C           CMP A  #',
14550 0B0B 27 CA           BEQ    INPUT1
14560 0B0D 09       INPEX  DEX
14570 0B0E 7F 0040         CLR    PRCNT
14580 0B11 81 1E           CMP A  #$1E
14590 0B13 26 03           BNE    DBLLTR
14600 0B15 7E 0DDF         JMP    REMARK
14610 0B18 C6 03    DBLLTR LDA B  #3
14620 0B1A 7E 0828         JMP    ERROR
```

```
14650 0B1D BD 0744 INNUM   JSR     SKIPSP
14660 0B20 97 4A           STA A   TSIGN
14670 0B22 08              INX
14680 0B23 81 2D           CMP A   #'-
14690 0B25 27 04           BEQ     INNUMO
14700 0B27 09              DEX

14720 0B28 7F 004A TSTN    CLR     TSIGN
14730 0B2B 4F      INNUMO  CLR A
14740 0B2C 97 4C           STA A   SIGDIG
14750 0B2E 97 4D           STA A   DECFLG
14760 0B30 97 4F           STA A   TOGGLE
14770 0B32 BD 0BDD         JSR     CLRACC
14780 0B35 D6 5D           LDA B   ASTK
14790 0B37 D7 26           STA B   INDEX4
14800 0B39 D6 5E           LDA B   ASTK+1
14810 0B3B D7 27           STA B   INDEX4+1
14820 0B3D C6 05           LDA B   #5
14830 0B3F BD 0744 INTST1  JSR     SKIPSP
14840 0B42 BD 0432         JSR     TESTNO
14850 0B45 24 0B           BCC     INTST2-1
14860 0B47 81 2E           CMP A   #'.
14870 0B49 26 05           BNE     *+7
14880 0B4B 97 4D           STA A   DECFLG
14890 0B4D 08              INX
14900 0B4E 20 EF           BRA     INTST1
14910 0B50 0D              SEC
14920 0B51 39              RTS
14930 0B52 09              DEX
14940 0B53 08      INTST2  INX
14950 0B54 A6 00           LDA A   0,X
14960 0B56 BD 0432         JSR     TESTNO
14970 0B59 25 42           BCS     MAYEXP
14980 0B5B 80 30           SUB A   #$30
14990 0B5D 26 0F           BNE     INTST3
15000 0B5F 7D 004C         TST     SIGDIG
15010 0B62 26 0C           BNE     INTST4
15020 0B64 7D 004D         TST     DECFLG
15030 0B67 27 FA           BEQ     INTST2
15040 0B69 7A 004B         DEC     EXP
15050 0B6C 20 E5           BRA     INTST2
15060 0B6E 97 4C   INTST3  STA A   SIGDIG
15070 0B70 BD 02B2 INTST4  JSR     PUSHX
15080 0B73 DF 26           LDX     INDEX4
15090 0B75 5D              TST B
15100 0B76 27 16           BEQ     INTST6
15110 0B78 7D 004F         TST     TOGGLE
15120 0B7B 26 08           BNE     INTST5
15130 0B7D AB 00           ADD A   0,X
15140 0B7F A7 00           STA A   0,X
15150 0B81 08              INX
15160 0B82 5A              DEC B
15170 0B83 20 06           BRA     INTS55
15180 0B85 48      INTST5  ASL A
15190 0B86 48              ASL A
15200 0B87 48              ASL A
15210 0B88 48              ASL A
15220 0B89 A7 00           STA A   0,X
15230 0B8B 73 004F INTS55  COM     TOGGLE
15240 0B8E DF 26   INTST6  STX     INDEX4
15250 0B90 7D 004D         TST     DECFLG
15260 0B93 26 03           BNE     INTS66
15270 0B95 7C 004B         INC     EXP
15280 0B98 BD 02C7 INTS66  JSR     PULLX
15290 0B9B 20 B6           BRA     INTST2
15300 0B9D 7D 004D MAYEXP  TST     DECFLG
15310 0BA0 26 08           BNE     MAYEX2
15320 0BA2 81 2E           CMP A   #'.
```

```
15330 0BA4 26 04          BNE    MAYEX2
15340 0BA6 97 4D          STA A  DECFLG
15350 0BA8 20 A9          BRA    INTST2
15360 0BAA 5F       MAYEX2 CLR B
15370 0BAB 81 45          CMP A  #'E
15380 0BAD 26 14          BNE    INNEX
15390 0BAF 08             INX
15400 0BB0 A6 00          LDA A  0,X
15410 0BB2 81 2D          CMP A  #'-
15420 0BB4 27 09          BEQ    INTST9
15430 0BB6 81 2B          CMP A  #'+
15440 0BB8 26 01          BNE    INTST8
15450 0BBA 08             INX
15460 0BBB 8D 35    INTST8 BSR    FORMEX
15470 0BBD 20 04          BRA    INNEX
15480 0BBF 08       INTST9 INX
15490 0BC0 8D 30          BSR    FORMEX
15500 0BC2 50             NEG B
15510 0BC3 96 4B    INNEX  LDA A  EXP
15520 0BC5 1B             ABA
15530 0BC6 BD 02B2        JSR    PUSHX
15540 0BC9 DE 5D          LDX    ARSTK
15550 0BCB A7 06          STA A  6,X
15560 0BCD BD 043F        JSR    PUSHAF
15570 0BD0 BD 02C7        JSR    PULLX
15580 0BD3 7D 004A        TST    TSIGN
15590 0BD6 27 03          BEQ    *+5
15600 0BD8 BD 051F        JSR    NEG
15610 0BDB 0C             CLC
15620 0BDC 39             RTS

15640 0BDD BD 02B2 CLRACC JSR    PUSHX
15650 0BE0 37             PSH B
15660 0BE1 C6 07          LDA B  #7
15670 0BE3 DE 5D          LDX    ARSTK
15680 0BE5 6F 00          CLR    0,X
15690 0BE7 08             INX
15700 0BE8 5A             DEC B
15710 0BE9 26 FA          BNE    *-4
15720 0BEB 7F 004B        CLR    EXP
15730 0BEE 33             PUL B
15740 0BEF 7E 02C7        JMP    PULLX

15760 0BF2 A6 00    FORMEX LDA A  0,X
15770 0BF4 BD 0432        JSR    TESTNO
15780 0BF7 24 05          BCC    *+7
15790 0BF9 C6 02    INNER  LDA B  #2
15800 0BFB 7E 0828        JMP    ERROR
15810 0BFE 80 30          SUB A  #$30
15820 0C00 16             TAB
15830 0C01 08             INX
15840 0C02 A6 00          LDA A  0,X
15850 0C04 BD 0432        JSR    TESTNO
15860 0C07 25 0D          BCS    *+15
15870 0C09 58             ASL B
15880 0C0A 17             TBA
15890 0C0B 58             ASL B
15900 0C0C 58             ASL B
15910 0C0D 1B             ABA
15920 0C0E 16             TAB
15930 0C0F A6 00          LDA A  0,X
15940 0C11 80 30          SUB A  #$30
15950 0C13 1B             ABA
15960 0C14 16             TAB
15970 0C15 08             INX
15980 0C16 39             RTS
```

```
16000 0C17 DE 34    READ   LDX    BASPNT
16010 0C19 BD 03C1  READ1  JSR    TSTV
16020 0C1C 25 3F           BCS    READEX
16030 0C1E DF 34           STX    BASPNT
16040 0C20 DE 3C           LDX    DATPNT
16050 0C22 9C 2F           CPX    SOURCE
16060 0C24 27 1C           BEQ    READ33
16070 0C26 BD 0B1D  READ2  JSR    INNUM
16080 0C29 24 22           BCC    READ4
16090 0C2B 09              DEX
16100 0C2C A6 00           LDA A  0,X
16110 0C2E 81 1E           CMP A  #$1E
16120 0C30 27 05           BEQ    READ3
16130 0C32 C6 19    RDERR  LDA B  #$19
16140 0C34 7E 0828         JMP    ERROR
16150 0C37 9C 2A    READ3  CPX    NEXTRA
16160 0C39 27 F7           BEQ    RDERR
16170 0C3B A6 00           LDA A  0,X
16180 0C3D 08              INX
16190 0C3E 81 1E           CMP A  #$1E
16200 0C40 26 F5           BNE    READ3
16210 0C42 A6 02    READ33 LDA A  2,X
16220 0C44 81 79           CMP A  #$79     VECTOR OF DATA STATEMENT
16230 0C46 26 EF           BNE    READ3
16240 0C48 08              INX
16250 0C49 08              INX
16260 0C4A 08              INX
16270 0C4B 20 D9           BRA    READ2
16280 0C4D BD 02E0  READ4  JSR    STORE
16290 0C50 08              INX
16300 0C51 DF 3C           STX    DATPNT
16310 0C53 DE 34           LDX    BASPNT
16320 0C55 BD 0744         JSR    SKIPSP
16330 0C58 08              INX
16340 0C59 81 2C           CMP A  #',
16350 0C5B 27 BC           BEQ    READ1
16360 0C5D 09       READEX DEX
16370 0C5E 81 1E           CMP A  #$1E
16380 0C60 26 D0           BNE    RDERR
16390 0C62 20 06           BRA    RESTO1



16410 0C64 DE 2F    RESTOR LDX    SOURCE
16420 0C66 DF 3C           STX    DATPNT
16430 0C68 DE 34           LDX    BASPNT
16440 0C6A 7E 0DDF  RESTO1 JMP    REMARK



16460 0C6D DE 34    PRINT  LDX    BASPNT
16470 0C6F BD 0744  PRINT0 JSR    SKIPSP
16480 0C72 81 22           CMP A  #'"
16490 0C74 26 14           BNE    PRINT4
16500 0C76 08              INX
16510 0C77 A6 00    PRINT1 LDA A  0,X
16520 0C79 08              INX
16530 0C7A 81 22           CMP A  #'"
16540 0C7C 27 4A           BEQ    PRINT8
16550 0C7E 81 1E           CMP A  #$1E
16560 0C80 26 04           BNE    PRINT2
16570 0C82 C6 04           LDA B  #4
16580 0C84 20 1C           BRA    PRINTF
16590 0C86 8D 6F    PRINT2 BSR    OUTCHA
```

```
16600 0C88 20 FD          BRA    PRINT1
16610 0C8A 81 1F   PRINT4 CMP A  #$1F
16620 0C8C 26 35          BNE    PRINT8
16630 0C8E 09             DEX
16640 0C8F A6 00          LDA A  0,X
16650 0C91 08             INX
16660 0C92 81 3B          CMP A  #';
16670 0C94 27 06          BEQ    PRINT5
16680 0C96 BD 02A0        JSR    CRLF
16690 0C99 7F 0040        CLR    PRCNT
16700 0C9C 08        PRINT5 INX
16710 0C9D DF 36          STX    BASLIN
16720 0C9F 7E 0A00        JMP    BASIC
16730 0CA2 7E 0828 PRINTE JMP    ERROR
16740 0CA5 DE 34   CHR    LDX    BASPNT
16750 0CA7 BD 0467        JSR    FIX
16760 0CAA 17      CHR2   TBA
16770 0CAB 8D 49          BSR    OUTCHA
16780 0CAD 8E A045 PRINT6 LDS    #$A045
16790 0CB0 20 16          BRA    PRIN88
16800 0CB2 DE 34   TAB    LDX    BASPNT
16810 0CB4 BD 0467        JSR    FIX
16820 0CB7 D0 40          SUB B  PRCNT
16830 0CB9 23 F2          BLS    PRINT6
16840 0CBB BD 0350 PRIN77 JSR    PRINSP
16850 0CBE 5A             DEC B
16860 0CBF 26 FA          BNE    PRIN77
16870 0CC1 20 FA          BRA    PRINT6
16880 0CC3 BD 04E6 PRINT8 JSR    EXPR
16890 0CC6 8D 4B          BSR    PRN
16900 0CC8 BD 0744 PRIN88 JSR    SKIPSP
16910 0CCB 81 2C          CMP A  #',
16920 0CCD 26 0E          BNE    PRIN99
16930 0CCF 08             INX
16940 0CD0 96 40   PRLOOP LDA A  PRCNT
16950 0CD2 16             TAB
16960 0CD3 C4 F0          AND B  #$F0
16970 0CD5 10             SBA
16980 0CD6 27 0A          BEQ    PRI999
16990 0CD8 BD 0350        JSR    PRINSP

17010 0CDB 20 F3          BRA    PRLOOP
17020 0CDD 81 3B   PRIN99 CMP A  #';
17030 0CDF 26 04          BNE    PREND
17040 0CE1 08             INX
17050 0CE2 7E 0C6F PRI999 JMP    PRINTO
17060 0CE5 81 1F   PREND  CMP A  #$1F
17070 0CE7 27 A1          BEQ    PRINT4
17080 0CE9 C6 06          LDA B  #6
17090 0CEB 20 B5          BRA    PRINTE

17110 0CED 86 2F   OUTPNT LDA A  #'.
17120 0CEF 20 05          BRA    OUTCHA
17130 0CF1 4F      OUTZER CLR A
17140 0CF2 84 0F   OUTDIG AND A  #$0F
17150 0CF4 8B 30          ADD A  #$30
17160 0CF6 36      OUTCHA PSH A
17170 0CF7 BD 0276        JSR    OUTCH
17180 0CFA 32             PUL A
```

```
17200 0CFB 37       ENLINE PSH B
17210 0CFC D6 40           LDA B  PRCNT
17220 0CFE 5C              INC B
17230 0CFF C1 30           CMP B  #48
17240 0D01 23 0C           BLS    ENLEXT
17250 0D03 C1 3F           CMP B  #63
17260 0D05 27 04           BEQ    *+6
17270 0D07 81 20           CMP A  #$20
17280 0D09 26 04           BNE    ENLEXT
17290 0D0B BD 02A0         JSR    CRLF
17300 0D0E 5F              CLR B
17310 0D0F D7 40    ENLEXT STA B  PRCNT
17320 0D11 33              PUL B
17330 0D12 39              RTS



17350 0D13 BD 0445 PRN     JSR    PSHXPA
17360 0D16 4F              CLR A
17370 0D17 97 4D           STA A  DECFLG
17380 0D19 6D 00           TST    0,X
17390 0D1B 26 08           BNE    PRN1
17400 0D1D 8D D2           BSR    OUTZER
17410 0D1F BD 02C7 PRN0    JSR    PULLX
17420 0D22 7E 0350         JMP    PRINSP
17430 0D25 6D 00    PRN1   TST    0,X
17440 0D27 2A 07           BPL    PRN2
17450 0D29 86 2D           LDA A  #'-
17460 0D2B 8D C9           BSR    OUTCHA
17470 0D2D BD 0528         JSR    NEGACC
17480 0D30 A6 06    PRN2   LDA A  6,X
17490 0D32 81 09           CMP A  #9
17500 0D34 2F 31           BGT    PRISCI
17510 0D36 81 FF           CMP A  #-1
17520 0D38 2D 2D           BLT    PRISCI
17530 0D3A 4D              TST A
17540 0D3B 2E 10           BGT    PRN4
17550 0D3D 8D B2           BSR    OUTZER
17560 0D3F 8D AC           BSR    OUTPNT
17570 0D41 97 4D           STA A  DECFLG
17580 0D43 6D 06    PRN3   TST    6,X
17590 0D45 2A 1C           BPL    PRN5
17600 0D47 8D A8           BSR    OUTZER
17610 0D49 6C 06           INC    6,X
17620 0D4B 20 F6           BRA    PRN3
17630 0D4D 7D 004D PRN4    TST    DECFLG
17640 0D50 26 11           BNE    PRN5
17650 0D52 A6 00    PRN44  LDA A  0,X
17660 0D54 8D 9C           BSR    OUTDIG
17670 0D56 BD 0552         JSR    ALLEFT
17680 0D59 6A 06           DEC    6,X
17690 0D5B 26 F5           BNE    PRN44
17700 0D5D 8D 45           BSR    TSTZER
17710 0D5F 24 BF           BCC    PRN0
17720 0D61 8D 8A           BSR    OUTPNT
17730 0D63 8D 30    PRN5   BSR    PRTAIL
17740 0D65 20 B8           BRA    PRN0
17750 0D67 A6 00    PRISCI LDA A  0,X
17760 0D69 8D 87           BSR    OUTDIG
17770 0D6B BD 0552         JSR    ALLEFT
17780 0D6E BD 0CED         JSR    OUTPNT
17790 0D71 8D 22           BSR    PRTAIL
17800 0D73 86 45           LDA A  #'E
17810 0D75 BD 0CF6         JSR    OUTCHA
17820 0D78 E6 06           LDA B  6,X
17830 0D7A 5A              DEC B
```

```
17840 0D7B 2A 06          BPL    PRISC2
17850 0D7D 86 2D          LDA A  #'-
17860 0D7F BD 0CF6        JSR    OUTCHA
17870 0D82 50             NEG B
17880 0D83 4F       PRISC2 CLR A
17890 0D84 4C       PRISC3 INC A
17900 0D85 C0 0A          SUB B  #10
17910 0D87 2A FB          BPL    PRISC3
17920 0D89 4A             DEC A
17930 0D8A CB 0A          ADD B  #10
17940 0D8C BD 0CF2        JSR    OUTDIG
17950 0D8F 17             TBA
17960 0D90 BD 0CF2        JSR    OUTDIG
17970 0D93 20 8A          BRA    PRNO

17990 0D95 8D 0D    PRTAIL BSR    TSTZER
18000 0D97 24 0A          BCC    PRTAEX
18010 0D99 A6 00          LDA A  0,X
18020 0D9B BD 0CF2        JSR    OUTDIG
18030 0D9E BD 0552        JSR    ALLEFT
18040 0DA1 20 F2          BRA    PRTAIL
18050 0DA3 39       PRTAEX RTS

18070 0DA4 BD 02B2 TSTZER JSR    PUSHX
18080 0DA7 DE 5D          LDX    AESTK
18090 0DA9 37             PSH B
18100 0DAA C6 06          LDA B  #6
18110 0DAC 6D 00    TSTZ1 TST    0,X
18120 0DAE 26 07          BNE    TSTZ3
18130 0DB0 08             INX
18140 0DB1 5A             DEC B
18150 0DB2 26 F8          BNE    TSTZ1
18160 0DB4 0C             CLC
18170 0DB5 20 01          BRA    TSTZ3+1
18180 0DB7 0D       TSTZ3 SEC
18190 0DB8 33             PUL B
18200 0DB9 7E 02C7        JMP    PULLX

18220 0DBC DE 34    LET   LDX    BASPNT
18230 0DBE BD 03C1        JSR    TSTV
18240 0DC1 24 05          BCC    *+7
18250 0DC3 C6 12    LET0  LDA B  #$12
18260 0DC5 7E 0828  LET00 JMP    ERROR
18270 0DC8 BD 0744        JSR    SKIPSP
18280 0DCB 08             INX
18290 0DCC 81 3D          CMP A  #'=
18300 0DCE 27 04          BEQ    *+6
18310 0DD0 C6 06    LET2  LDA B  #6
18320 0DD2 20 F1          BRA    LET00
18330 0DD4 BD 04F6        JSR    EXPR
18340 0DD7 81 1E          CMP A  #$1E
18350 0DD9 26 F5          BNE    *-9
18360 0DDB BD 02F0        JSR    STORE

18380 0DDE BD 077D REMARK JSR    NXTLIN
18390 0DE1 7E 0A00        JMP    BASIC
```

```
18410 0DE4 BD 02B2 GETPAR JSR    PUSHX
18420 0DE7 DE 34          LDX    BASPNT
18430 0DE9 BD 04E6        JSR    EXPR
18440 0DEC 08             INX
18450 0DED 81 2C          CMP A  #',
18460 0DEF 27 0B          BEQ    GETPA2
18470 0DF1 81 29          CMP A  #')
18480 0DF3 26 1A          BNE    DIMERR
18490 0DF5 DF 34          STX    BASPNT
18500 0DF7 BD 02C7        JSR    PULLX
18510 0DFA 0C             CLC
18520 0DFB 39             RTS
18530 0DFC BD 04B1 GETPA2 JSR    PAREXP
18540 0DFF DF 34          STX    BASPNT
18550 0E01 BD 02C7        JSR    PULLX
18560 0E04 0D             SEC
18570 0E05 39             RTS
```

```
18590 0E06 DF 34    DIM     LDX    BASPNT
18600 0E08 BD 0358          JSR    FINVAR
18610 0E0B 29 02            BVS    DIMERR
18620 0E0D 25 05            BCS    DIM1
18630 0E0F C6 05    DIMERR  LDA B  #5
18640 0E11 7E 0828          JMP    ERROR
18650 0E14 81 28    DIM1    CMP A  #'(
18660 0E16 26 F7            BNE    DIMERR
18670 0E18 DF 34            STX    BASPNT
18680 0E1A DE 63            LDX    VARPNT
18690 0E1C E7 00            STA B  0,X
18700 0E1E A7 01            STA A  1,X
18710 0E20 8D C2            BSR    GETPAR
18720 0E22 24 05            BCC    DIM3
18730 0E24 BD 0467          JSR    FIX
18740 0E27 20 01            BRA    DIM4
18750 0E29 5F       DIM3    CLR B
18760 0E2A E7 03    DIM4    STA B  3,X
18770 0E2C BD 0467          JSR    FIX
18780 0E2F 17               TBA
18790 0E30 A7 02            STA A  2,X
18800 0E32 8D 0D            BSR    DIMCAL
18810 0E34 DF 34            LDX    BASPNT
18820 0E36 BD 0744          JSR    SKIPSP
18830 0E39 08               INX
18840 0E3A 81 2C            CMP A  #',
18850 0E3C 27 CA            BEQ    DIM+2
18860 0E3E 09               DEX
18870 0E3F 20 9D            BRA    REMARK
```

```
18890 0E41 BD 02B2 DIMCAL JSR    PUSHX
18900 0E44 E6 03          LDA B  3,X
18910 0E46 8D 11          BSR    SUBCAL
18920 0E48 BD 0459        JSR    INXSIX
18930 0E4B DF 46          STX    ARRTAB
18940 0E4D BD 02C7        JSR    PULLX
18950 0E50 96 46          LDA A  ARRTAB
18960 0E52 A7 04          STA A  4,X
18970 0E54 96 47          LDA A  ARRTAB+1
18980 0E56 A7 05          STA A  5,X
18990 0E58 39             RTS
```

```
19010 0E59 4D       SUBCAL TST A
19020 0E5A 26 05           BNE    SUBC1
19030 0E5C C6 15    SUBERR LDA B  #$15
19040 0E5E 7E 0828         JMP    ERROR
19050 0E61 A1 02    SUBC1  CMP A  2,X

19060 0E63 22 F7           BHI    SUBERR
19070 0E65 E1 03           CMP B  3,X
19080 0E67 22 F3           BHI    SUBERR
19090 0E69 36              PSH A
19100 0E6A A6 02           LDA A  2,X
19110 0E6C 97 51           STA A  ANUMB
19120 0E6E 5D              TST B
19130 0E6F 27 0D           BEQ    SUBC6
19140 0E71 5A       SUBC4  DEC B
19150 0E72 27 0A           BEQ    SUBC6
19160 0E74 96 51           LDA A  ANUMB
19170 0E76 BD 0459 SUBC5   JSR    INXSIX
19180 0E79 4A              DEC A
19190 0E7A 26 FA           BNE    SUBC5
19200 0E7C 20 F3           BRA    SUBC4
19210 0E7E 32       SUBC6  PUL A
19220 0E7F BD 0459 SUBC7   JSR    INXSIX
19230 0E82 4A              DEC A
19240 0E83 26 FA           BNE    SUBC7
19250 0E85 39              RTS



19270 0E86 DE 34    FOR    LDX    BASPNT
19280 0E88 BD 03C1         JSR    TSTV
19290 0E8B 24 03           BCC    FOR1
19300 0E8D 7E 0DC3         JMP    LET0      ERROR JUMP
19310 0E90 DF 34    FOR1   STX    BASPNT
19320 0E92 DE 5D           LDX    AESTK
19330 0E94 BD 044F         JSR    DEXSEV
19340 0E97 A6 00           LDA A  0,X
19350 0E99 E6 01           LDA B  1,X
19360 0E9B 97 51           STA A  ANUMB
19370 0E9D D7 52           STA B  BNUMB
19380 0E9F CE 1180         LDX    #FORSTK
19390 0EA2 96 51    FOR10  LDA A  ANUMB
19400 0EA4 D6 52           LDA B  BNUMB
19410 0EA6 9C 5F           CPX    FORPNT
19420 0EA8 27 1C           BEQ    FOR14
19430 0EAA A1 00           CMP A  0,X
19440 0EAC 26 10           BNE    FOR12
19450 0EAE E1 01           CMP B  1,X
19460 0EB0 26 0C           BNE    FOR12
19470 0EB2 96 36           LDA A  BASLIN
19480 0EB4 D6 37           LDA B  BASLIN+1
19490 0EB6 A1 0E           CMP A  14,X
19500 0EB8 26 04           BNE    FOR12
19510 0EBA E1 0F           CMP B  15,X
19520 0EBC 27 08           BEQ    FOR14
19530 0EBE BD 0457 FOR12   JSR    INXATE
19540 0EC1 BD 0457         JSR    INXATE
19550 0EC4 20 DC           BRA    FOR10
19560 0EC6 8C 1200 FOR14   CPX    #FORSTK+128
19570 0EC9 26 05           BNE    FOR11
19580 0ECB C6 10           LDA B  #16
19590 0ECD 7E 0828         JMP    ERROR
19600 0ED0 96 51    FOR11  LDA A  ANUMB
19610 0ED2 D6 52           LDA B  BNUMB
19620 0ED4 A7 00           STA A  0,X
19630 0ED6 08              INX
```

```
19640 0ED7 E7 00          STA B  0,X
19650 0ED9 08             INX
19660 0EDA DF 5F          STX    FORPNT
19670 0EDC DE 34          LDX    BASPNT
19680 0EDE BD 0744        JSR    SKIPSP
19690 0EE1 08             INX
19700 0EE2 81 3D          CMP A  #'=
19710 0EE4 27 03          BEQ    FOR3
19720 0EE6 7E 0DD0 FOR2   JMP    LET2       ERROR JUMP
19730 0EE9 BD 04E6 FOR3   JSR    EXPR
19740 0EEC BD 02F0        JSR    STORE
19750 0EEF BD 0792        JSR    CCINT
19760 0EF2 8C 0109        CPX    #TOPNT
19770 0EF5 26 FF          BNE    FOR2
19780 0EF7 DE 34          LDX    BASPNT
19790 0EF9 BD 04E6        JSR    EXPR
19800 0EFC DF 34          STX    BASPNT
19810 0EFE DE 5F          LDX    FORPNT
19820 0F00 BD 02D2        JSR    STOREX
19830 0F03 BD 0459        JSR    INXSTX
19840 0F06 DF 5F          STX    FORPNT
19850 0F08 DE 34          LDX    BASPNT
19860 0F0A BD 0792        JSR    CCINT
19870 0F0D 8C 0110        CPX    #STEPNT
19880 0F10 26 09          BNE    FOR4
19890 0F12 DF 34          LDX    BASPNT
19900 0F14 BD 04E6        JSR    EXPR
19910 0F17 DF 34          STX    BASPNT
19920 0F19 20 0F          BRA    FOR5
19930 0F1B DE 5D   FOR4   LDX    AESTK
19940 0F1D BD 0BDD        JSR    CLRACC
19950 0F20 86 01          LDA A  #$01
19960 0F22 A7 00          STA A  0,X
19970 0F24 A7 06          STA A  6,X
19980 0F26 BD 043E        JSR    PUSHAE
19990 0F29 DE 5F   FOR5   LDX    FORPNT
20000 0F2B BD 02D2        JSR    STOREX
20010 0F2E BD 0459        JSR    INXSTX
20020 0F31 DF 5F          STX    FORPNT
20030 0F33 DE 34   FOR7   LDX    BASPNT
20040 0F35 BD 0744        JSR    SKIPSP
20050 0F38 81 1F          CMP A  #$1F
20060 0F3A 26 AA   FOR8   BNE    FOR2       ERROR JUMP
20070 0F3C 08             INX
20080 0F3D DF 36          STX    BASLIN
20090 0F3F DE 5F          LDX    FORPNT
20100 0F41 D6 36          LDA B  BASLIN
20110 0F43 E7 00          STA B  0,X
20120 0F45 08             INX
20130 0F46 D6 37          LDA B  BASLIN+1
20140 0F48 E7 00          STA B  0,X
20150 0F4A 08             INX
20160 0F4B DF 5F          STX    FORPNT
20170 0F4D 7E 0A00        JMP    BASIC



20190 0F50 DE 34   NEXT   LDX    BASPNT
20200 0F52 BD 03C1        JSR    TSTV
20210 0F55 24 03          BCC    NEXT1
20220 0F57 7E 0DC3        JMP    LET0       ERROR JUMP
20230 0F5A BD 0744 NEXT1  JSR    SKIPSP
20240 0F5D 81 1F          CMP A  #$1F
20250 0F5F 26 D9          BNE    FOR8       ERROR JUMP
20260 0F61 08             INX
20270 0F62 DF 36          STX    BASLIN
```

```
20280 0F64 BD 0448          JSR    PULLAF
20290 0F67 A6 00            LDA A  0,X
20300 0F69 E6 01            LDA B  1,X
20310 0F6B DE 5F            LDX    FORPNT
20320 0F6D 8C 1180          CPX    #FORSTK
20330 0F70 27 66            BEQ    NEXT6
20340 0F72 DF 5F    NEXT2   STX    FORPNT
20350 0F74 8C 1180          CPX    #FORSTK
20360 0F77 27 5B            BEQ    NEXT5
20370 0F79 BD 044F          JSR    DEXATF
20380 0F7C BD 044F          JSR    DEXATF
20390 0F7F A1 00            CMP A  0,X
20400 0F81 26 EF            BNE    NEXT2
20410 0F83 E1 01            CMP B  1,X
20420 0F85 26 EB            BNE    NEXT2
20430 0F87 DF 61            STX    FORNOW
20440 0F89 EE 00            LDX    0,X
20450 0F8B BD 0302          JSR    INDX
20460 0F8E DE 61            LDX    FORNOW
20470 0F90 BD 0457          JSR    INXATF
20480 0F93 86 02            LDA A  #2
20490 0F95 6D 00            TST    0,X
20500 0F97 2A 02            BPL    *+4
20510 0F99 8B 03            ADD A  #3
20520 0F9B 97 4F            STA A  NCMPR
20530 0F9D BD 0302          JSR    INDX
20540 0FA0 BD 05FC          JSR    ADD
20550 0FA3 CE 006F          LDX    #FORTMP
20560 0FA6 BD 02D2          JSR    STORFX
20570 0FA9 BD 043F          JSR    PUSHAF
20580 0FAC DE 61            LDX    FORNOW
20590 0FAE 08               INX
20600 0FAF 08               INX
20610 0FB0 BD 0302          JSR    INDX
20620 0FB3 BD 1037          JSR    CMPR
20630 0FB6 24 07            BCC    NEXT4
20640 0FB8 DE 61            LDX    FORNOW
20650 0FBA DF 5F            STX    FORPNT
20660 0FBC 7E 0A00  NEXT3   JMP    BASIC
20670 0FBF CE 006F  NEXT4   LDX    #FORTMP
20680 0FC2 BD 0302          JSR    INDX
20690 0FC5 DE 61            LDX    FORNOW
20700 0FC7 EE 00            LDX    0,X
20710 0FC9 BD 02D2          JSR    STORFX
20720 0FCC DE 61            LDX    FORNOW
20730 0FCE EE 0E            LDX    14,X
20740 0FD0 DF 36            STX    BASLIN
20750 0FD2 20 E8            BRA    NEXT3    EXIT
20760 0FD4 C6 18    NEXT5   LDA B  #$18
20770 0FD6 20 02            BRA    NEXT6+2
20780 0FD8 C6 17    NEXT6   LDA B  #$17
20790 0FDA 7E 0828          JMP    ERROR


20810 0FDD DF 34    IF      LDX    BASPNT
20820 0FDF BD 04E6          JSR    EXPR
20830 0FE2 8D 17            BSR    RELOP
20840 0FE4 97 4F            STA A  NCMPR
20850 0FE6 BD 04E6          JSR    EXPR
20860 0FE9 DF 34            STX    BASPNT
20870 0FEB 8D 4A            BSR    CMPR
20880 0FED 24 03            BCC    IF2
20890 0FEF 7E 0DDF          JMP    REMARK
20900 0FF2 DF 34    IF2     LDX    BASPNT
20910 0FF4 BD 079B          JSR    CCODE
20920 0FF7 EE 00            LDX    0,X
20930 0FF9 6E 00            JMP    0,X
```

```
20950 0FFB BD 0744 RELOP   JSR     SKIPSP
20960 0FFE 08              INX
20970 0FFF 81 3D           CMP A   #'=
20980 1001 26 03           BNE     RELOP0
20990 1003 86 00           LDA A   #0          =
21000 1005 39              RTS
21010 1006 E6 00    RELOP0 LDA B   0,X
21020 1008 81 3C           CMP A   #'<
21030 100A 26 13           BNE     RELOP4
21040 100C C1 3D           CMP B   #'=
21050 100E 26 04           BNE     RELOP1
21060 1010 08              INX
21070 1011 86 02           LDA A   #2          <=
21080 1013 39              RTS
21090 1014 C1 3E    RELOP1 CMP B   #'>
21100 1016 26 04           BNE     RELOP3
21110 1018 08       RELOP2 INX
21120 1019 86 03           LDA A   #3          <>
21130 101B 39              RTS
21140 101C 86 01    RELOP3 LDA A   #1          <
21150 101E 39              RTS
21160 101F 81 3E    RELOP4 CMP A   #'>
21170 1021 27 05           BEQ     REL44
21180 1023 C6 06           LDA B   #6
21190 1025 7E 0828         JMP     ERROR
21200 1028 C1 3D    REL44  CMP B   #'=
21210 102A 26 04           BNE     RELOP5
21220 102C 08              INX
21230 102D 86 05           LDA A   #5          >=
21240 102F 39              RTS
21250 1030 C1 3C    RELOP5 CMP B   #'<
21260 1032 27 F4           BEQ     RELOP2
21270 1034 86 04           LDA A   #4          >
21280 1036 39              RTS



21300 1037 96 4F    CMPR   LDA A   NOCMPR
21310 1039 48              ASL A
21320 103A 48              ASL A
21330 103B CE 1051         LDX     #CMPR1
21340 103E DF 20           STX     INDEX1
21350 1040 9B 21           ADD A   INDEX1+1
21360 1042 97 21           STA A   INDEX1+1
21370 1044 BD 05F9         JSR     SUB
21380 1047 BD 0448         JSR     PULLAF
21390 104A A6 00           LDA A   0,X
21400 104C DF 20           LDX     INDEX1
21410 104E 4D              TST A
21420 104F 6F 00           JMP     0,X
21430              *   ******FOLLOWING MUST BE ON ONE PAGE******
21440 1051 27 16    CMPR1  BEQ     OKCMPR
21450 1053 20 12           BRA     NOCMPR
21460 1055 2B 12           BMI     OKCMPR
21470 1057 20 0E           BRA     NOCMPR
21480 1059 2B 0E           BMI     OKCMPR
21490 105B 20 F4           BRA     CMPR1
21500 105D 26 0A           BNE     OKCMPR
21510 105F 20 06           BRA     NOCMPR
21520 1061 27 04           BEQ     NOCMPR
21530 1063 2B 02           BMI     NOCMPR
21540 1065 2A 02           BPL     OKCMPR
21550 1067 0D       NOCMPR SEC
21560 1068 39              RTS
21570 1069 0C       OKCMPR CLC
21580 106A 39              RTS
```

```
21600 106B BD 0448 USER    JSR     PULLAE
21610 106E BD 02B2         JSR     PUSHX
21620 1071 DF 67           LDX     DUMMY
21630 1073 AD 00           JSR     0,X
21640 1075 BD 02C7         JSR     PULLX

21660 1078 39        DUMRTS RTS



21680 1079 BD 0448 INT     JSR     PULLAE
21690 107C C6 09   INT1    LDA B   #9
21700 107E F0 06           SUB B   6,X
21710 1080 5D      INT2    TST B
21720 1081 2F 08           BLE     INT4
21730 1083 BD 0578         JSR     LRIGHT
21740 1086 6C 06           INC     6,X
21750 1088 5A              DEC B
21760 1089 20 F5           BRA     INT2
21770 108B 6F 05   INT4    CLR     5,X
21780 108D BD 059F         JSR     NORMPA
21790 1090 7E 0448         JMP     PULLAE



21810 1093 BD 0448 SGN     JSR     PULLAE
21820 1096 A6 00           LDA A   0,X
21830 1098 27 DF           BEQ     DUMRTS
21840 109A 97 4A           STA A   TSIGN
21850 109C BD 0BDD         JSR     CLRACC
21860 109F 86 01           LDA A   #1
21870 10A1 A7 00           STA A   0,X
21880 10A3 A7 06           STA A   6,X
21890 10A5 7D 004A SGN3    TST     TSIGN
21900 10A8 2A CF           BPL     DUMRTS
21910 10AA 7E 0528 SGN33   JMP     NEGACC
21920 10AD 39      SGN4    RTS

21940 10AE BD 0448 ABS     JSR     PULLAE
21950 10B1 6D 00           TST     0,X
21960 10B3 2A C3           BPL     DUMRTS
21970 10B5 20 F3           BRA     SGN33



21990 10B7 BD 0448 RANDOM  JSR     PULLAE
22000 10BA BD 0DA4         JSR     TSTZER
22010 10BD 24 05           BCC     RAND1
22020 10BF BD 043F         JSR     PUSHAE
22030 10C2 20 08           BRA     RAND1+8
22040 10C4 CE 0069 RAND1   LDX     #RANNUM
22050 10C7 6C 00           INC     0,X
22060 10C9 BD 0302         JSR     INDX
22070 10CC CE 10EE         LDX     #RANMUL
22080 10CF BD 0302         JSR     INDX
22090 10D2 BD 06D1         JSR     MPY
22100 10D5 BD 0448         JSR     PULLAE
22110 10D8 BD 0552         JSR     ALLEFT
22120 10DB E6 01           LDA B   1,X
22130 10DD A6 03           LDA A   3,X
22140 10DF A7 01           STA A   1,X
```

```
22150 10E1 E7 03          STA B  3,X
22160 10E3 6F 06          CLR    6,X
22170 10E5 BD 059F        JSR    NORMPA
22180 10E8 CF 0069        LDX    #RANNUM
22190 10EB 7F 02D2        JMP    STOREX
22200 10EE 08        RANMUL FCB    $08,$37,$25,$41,$69,$00
      10EF 37
      10F0 25
      10F1 41
      10F2 69
      10F3 00
22210      10F4    ROBERT EQU    *



22230 1102                ORG    $1102
22240 1102 007F     ASTACK RMB    18*7
22250 1180 0080     FORSTK RMB    128
22260 1200                ORG    $1200
22270      1200     FILE   EQU    *
22280 A048                ORG    $A048
22290 A048 0100           FDB    PROGM
22300                     END
```

SYMBOL TABLE

```
INDEX1 0020  INDEX2 0022  INDEX3 0024  INDEX4 0026  SAVESP 0028
NEXTBA 002A  WORKBA 002C  SOURCE 002E  PACKLN 0030  HIGHLN 0032
BASPNT 0034  BASLIN 0036  PUSHTX 0038  XSTACK 003A  DATPNT 003C
DIMPNT 003F  PRCNT  0040  MEMEND 0044  ARRTAB 0046  KEYWD  0048
TSIGN  004A  EXP    004B  SIGDIG 004C  DECFLG 004D  TOGGLE 004E
NCMPR  004F  TNUMB  0050  ANUMB  0051  BNUMB  0052  LRNINE 0053
ROUNDV 0054  ONLOOP 0055  NORSGN 0056  XLONG  0057  MPYXX  0059
BASNUM 005B  AESTK  005D  FORPNT 005F  FORNOW 0061  VARPNT 0063
SBRPNT 0065  DUMMY  0067  RANNUM 0069  FORTMP 006F  SBRSTK 0075
LOAFLG 0085  BUFNXT 00AC  ENDBUF 00AE  BUFFER 00B0  PROGM  0100
INTER  0106  TOPNT  0109  STEPNT 0110  COMMAN 0112  GOLIST 0137
STOMSG 01B3  COMEND 01BA  IMPLET 01BB  FUNCT  01BD  FUNEND 01ED
RDYMSG 01F0  DELMSG 01FB  PGCNTL 0200  WHAT   0203  ERRMS1 0209
ERRMS2 0214  DEL    021E  KEYRD  0225  KEYRD0 022C  KEYRD1 022E
KEYBD2 0258  CNTLIN 025D  TEXIT  0263  OUT2H  0270  OUT4HS 0273
OUTCH  0276  INCH   027B  BREAK  0281  BREAK0 0289  BREAK1 0290
OUTPUT 0292  OUTNCR 0299  CRLF   02A0  CRLFST 02AA  CREND  02B1
PUSHX  02B2  PULLX  02C7  STOREX 02D2  STORE  02E0  INDX   0302
IND    0309  INDO   0313  OUTLIN 032A  PRINSP 0350  FINVAR 0358
FINV1  0360  FINV11 0386  FINV2  0388  FINV3  0394  FINV4  0398
FINV5  03A9  FINV6  03B0  FINV7  03BD  TSTV   03C1  TSTV1  03C9
TSTV11 03D4  TSTV2  03D6  TSTV3  03E9  TSTV31 03EE  TSTV33 03F2
TSTV34 03F3  TSTV4  03FD  TSTV44 0417  TSTV5  0422  TSTLTR 042A
TESTNO 0432  NONO   043A  YESNO  043C  PUSHAF 043E  PSHXPA 0445
PULLAE 0448  DEXATF 044E  DEXSEV 044E  DEXSIX 0450  DEXFIV 0451
DEXFOR 0452  INXATF 0457  INXSEV 0458  INXSIX 0459  INXFIV 045A
INXFOR 045B  OVRTST 045F  FIX    0467  FIXERR 0470  FIX2   0477
FIXEX  0494  FACT   049A  PAREXP 04B1  FACT2  04BC  FACT3  04C1
TERM   04C5  TERM0  04C7  TERM1  04D9  TERM2  04E5  EXPR   04E6
EXPR1  04FB  NEGOP  0517  NEG    051E  NEGACC 0528  NEGA11 0532
NEGA2  0542  ALLEFT 0552  LLEFT  0550  LLEFT2 0563  LLEFT3 056C
LRIGHT 0578  LRIGH2 0584  LRIGH3 058A  NORMPA 059E  NORM   05A1
NORM00 05B5  NORM1  05BB  NORMEZ 05C5  NORMEX 05C7  NORM01 05D4
NORMEO 05E4  NORMF1 05EB  NORMF2 05F3  SUB    05F9  ADD    05FC
ADDO   0611  ADD1   0622  ADD2   0624  ADD4   0629  SWAP   062C
SWAP1  0632  SWAP2  0640  ADDER  0645  ADDER1 064C  DIV    0659
DIV2   0678  DIV3   0679  MDSIGN 06A8  MDS2   06B5  MDS3   06C1
MDS4   06CF  MPY    06D1  MPY4   06F3  MPY5   06FA  MPY6   06F3
MDEXIT 06FE  MDFX2  0713  FINDNO 0716  FINDN1 0720  FINDO  0722
HIBALL 073E  SKIPSP 0744  INXSKP 074A  LINENO 074F  LINEO  0753
NXTLIN 077D  FCODE  0789  CCINT  0792  CCODE  079B  LOOP3  07A4
LOOP5  07B6  CCEXIT 07D3  FUNCEX 07D9  NEWSUB 07DC  START  07E2
READY  07F6  NEWLIN 0807  ERROR  0828  ERROR2 0853  RUN    085B
CLIST  0882  CLIST3 08A8  CLIST4 08BC  CLIST5 08C2  CLEXT  08D0
PATCH  08D3  SAVE   08F5  SAVE1  08F2  SAVE5  0903  LOAD   090D
APPEND 0910  LOAD1  0917  DELAY2 0933  DELAY  0935  DELAY3 0937
LEADER 093E  NUMBER 0949  NUM1   094C  DELREP 0964  REPLAC 0977
```

```
NEXIT  097B  CAPPEN 097C  DELETE 0984  DEL2   098E  DEL4   0990
DELEX  09A7  INSERT 09AA  INS2   09D2  BUFWRT 09DC  OVERFL 09E8
BASIC  0A00  THEN   0A29  ONERR  0A36  ONGO   0A3B  GOSUB  0A5A
GOSUB1 0A6E  GOTO   0A7C  GOTO1  0A81  GOTO2  0A9D  RETURN 0AA2
STOP   0AB6  INPUT  0AC4  INPERR 0ACA  INPUT1 0AD7  INPUT2 0AE0
INPUT4 0AF6  INPUT5 0B01  INPFX  0B0D  DRLITR 0B18  INNUM  0B1D
TSTN   0B28  INNUM0 0B2B  INTST1 0B3E  INTST2 0B53  INTST3 0B6E
INTST4 0B70  INTST5 0B85  INTS55 0B8B  INTST6 0B8E  INTS66 0B98
MAYEXP 0B9D  MAYEX2 0BAA  INTST8 0BBB  INTST9 0BBE  INNEX  0BC3
CLRACC 0BDD  FORMFX 0BE2  INNER  0BE9  READ   0C17  READ1  0C19
READ2  0C26  RDERR  0C32  READ3  0C37  READ33 0C42  READ4  0C4D
READEX 0C5D  RESTOR 0C64  RESTO1 0C6A  PRINT  0C6D  PRINT0 0C6E
PRINT1 0C77  PRINT2 0C86  PRINT4 0C8A  PRINT5 0C9C  PRINTE 0CA2
CHR    0CA5  CHR2   0CAA  PRINT6 0CAD  TAB    0CB2  PRIN77 0CBB
PRINTB 0CC3  PRIN88 0CC8  PRLOOP 0CD0  PRIN99 0CDD  PRI999 0CE2
PREND  0CE5  OUTPNT 0CED  OUTZER 0CF1  OUTDIG 0CF2  OUTCHA 0CF6
ENLINE 0CFB  ENLEXT 0D0F  PRN    0D13  PRN0   0D1E  PRN1   0D25
PRN2   0D30  PRN3   0D43  PRN4   0D4D  PRN44  0D52  PRN5   0D63
PRISC1 0D67  PRISC2 0D83  PRISC3 0D84  PRTAIL 0D95  PRTAFX 0DA3
TSTZER 0DA4  TSTZ1  0DAC  TSTZ3  0DB7  LET    0DBC  LET0   0DC3
LET00  0DC5  LET2   0DD0  REMARK 0DDE  GETPAR 0DE4  GETPA2 0DEC
DIM    0E06  DIMERR 0E0E  DIM1   0E14  DIM3   0E29  DIM4   0E2A
DIMCAL 0E41  SUBCAL 0E59  SUBERR 0E5C  SUBC1  0E61  SUBC4  0E71
SUBC5  0E76  SUBC6  0E7E  SUBC7  0E7F  FOR    0E86  FOR1   0E90
FOR10  0EA2  FOR12  0EBE  FOR14  0EC6  FOR11  0ED0  FOR2   0EE6
FOR3   0EE9  FOR4   0F18  FOR5   0F29  FOR7   0F33  FOR8   0F3A
NEXT   0F50  NEXT1  0F5A  NEXT2  0F72  NEXT3  0F8C  NEXT4  0F8E
NEXT5  0FD4  NEXT6  0FD8  IF     0FDD  IF2    0FE2  RELOP  0FEB
RELOP0 1006  RELOP1 1014  RELOP2 1018  RELOP3 101C  RELOP4 101E
REL44  1028  RELOP5 1030  CMPR   1037  CMPR1  1051  NOCMPR 1067
OKCMPR 1069  USER   106B  DUMRTS 1078  INT    1079  INT1   107C
INT2   1080  INT4   108B  SGN    1093  SGN3   10A5  SGN33  10AA
SGN4   10AD  ABS    10AF  RANDOM 10B7  RAND1  10C4  RANMUL 10FF
ROBERT 10F4  ASTACK 1102  FORSTK 1180  FILE   1200

READY
```

UNDERSTANDING UITERWYK'S BASIC

Introduction

The following Table 1, SUMMARY OF BASIC FEATURES, explains the characteristics of the 6800 4K BASIC Interpreter program. For convenience Robert Uiterwyk's 4K BASIC also carries the mnemonic name RU4KBASIC.

BASIC Program Format

A BASIC program is comprised of programming statements. These statements tell the computer in a step-by-step sequence how to perform a particular task. The computer has no intelligence of its own, and only understands what it has been programmed to do. The BASIC language has some simple but exacting rules on how to instruct a computer how to do a particular task. For instance, the format of a BASIC programming statement is always a statement number, followed by the statement body, and terminated by a carriage return. The statement body always starts with a keyword that identifies the type of statement.

Statements need not be entered in numerical order, because the BASIC interpreter will automatically sort the statements in ascending order by line numbers. The statement number is also used for reference purposes, as will be seen when discussing the GOTO, GOSUB and IF statements. All line numbers must be between 1 and 9999, and zero may not be used. A programming statement may con-

Table 1. Summary of BASIC Features

COMMANDS	*STATEMENTS*		*FUNCTIONS*
LIST	REM	END	ABS
RUN	DIM	GOTO*	INT
NEW	DATA	ON...GOTO*	RND
SAVE	READ	ON...GOSUB*	SGN
LOAD	RESTORE	IF...THEN*	CHR
PATCH	LET*	INPUT	USER
	FOR	PRINT*	TAB
	NEXT	PATCH*	
	STOP	RETURN	
	GOSUB*		

*Flags statements that may be used in the direct mode (No statement numbers)

MATH OPERATORS	*RELATIONAL OPERATORS*
– Unary negation	= Equal
* Multiplication	<> NOT equal
/ Division	< Less than
+ Addition	> Greater than
– Subtraction	<= Less than or Equal
	>= Greater than or Equal

Line Numbers may be from 0001 to 9999

Variables	Simple variables	Single alphabetic or Single alphabetic and a single digit Single alphabetic
	Arrays:	
Backspace	Control-O	
Line delete	Control-X	
Panic Button	Control-C should bring Basic back to the READY mode regardless of what the BASIC user program is doing.	

tain no more than 72 characters including blanks. Unless within a character string and enclosed by quotation marks ("), blanks are *not* processed by BASIC, and their use is optional. With blanks, the statement is more readable, as can be seen in Example 1, but will require a slightly longer time to process. In addition typing blanks in the statements will require more memory for the storage of the program. Keep in mind, though, that program readability is very important; for if you cannot read it, you cannot modify or fix it! The only place in BASIC that blanks may not be used at all, is inside the various keywords and inside numbers. "12 34" is *NOT* the same as "1234".

In any one program, a line number may be used only once. A previously entered line may be changed by entering the same line

Programming Statement	Memory required
110 LET A = B + (3.5*5E2)	23 Bytes
110LETA = B + (3.5*5E2)	16 Bytes

Example 1. Statements with and without imbedded blanks.

number along with the desired new statement. Typing just a line number followed by a carriage return will delete that line.

It is strongly suggested that you use line numbers which are at least ten numbers apart. This will make it easier to add in statements between the original statements in case of omission, or if you desire to add additional features to an existing program. If you do wish to insert a statement between two others, you need only type a line number that falls between the other two. For example consider the following original BASIC program:

```
180 LET P = A*A*3.14
185 PRINT
```

in which it is desired to insert the following addition program statement

```
183 REM INSERT THIS LINE
```

with the resulting program becoming:

```
180 LET P = A*A*3.14
183 REM INSERT THIS LINE
185 PRINT
```

If it is desired to replace a statement, a new statement is typed that has the same line number as the one to be replaced. For example, typing the statement:

```
180P = (A*A)*3.14
```

would cause the example program to become:

```
180P = (A*A)*3.14
183 REM INSERT THIS LINE
185 PRINT
```

Each line is terminated by a carriage return. Only after the carriage return is typed will BASIC store the new statement in the proper sequence in its memory. If after typing a line and before typing the carriage return, you should change your mind, simply enter a "control-X" (Control key depressed and X key struck) to delete the line. Single character typing errors can be corrected by typing a "control-O" for a backspace. The computer will print either a left-arrow (←) or an underline (_) for each "control-O" pressed, and will backspace across the characters stored in the input buffer. The character displayed in response to a "control-O" will vary depending on the terminal used. If it is desired to stop a running program, or to terminate a LIST command, you should use the "control-C" character.

Variable Names And Data Formats

Now that you know how to enter a program, correct any mistyped lines, and how to stop a running program, we should probably find out how the "numbers" that control the program flow are referenced.

All numbers which can be changed during the execution of a program are called *variables*. Those numbers whose values cannot be changed other than by retyping a programming statement are called *constants*. Variables are referenced by a unique name given them by the programmer. A simple variable may be given a name of a single alphabetic *or* a single alphabetic character followed by a single digit 0 thru 9. Thus legal names for a simple variable are: A, B, Z0, Q5. Illegal names are 9, 5A, A34, AB. BASIC also has a type of variable known as an array. An array is defined as an ordered collection of numeric data known to BASIC under a single name. Any array may be given a name of a single alphabetic character, A to Z. BASIC allows the use of both the simple variable A and the array A in the same program.

Arrays are divided into columns (vertical) and rows (horizontal). Arrays may have one or two dimensions. For example:

1.01
2.11
3.22
4.34

is a one dimensional array while

6	5	4
3	2	1
0	9	8

is a two dimensional array.

Array elements are referenced by their column and row position. For instance, if the examples above were arrays A and Z respectively, 2.11 would be A(2); similarly, 0 would be Z(3,1). The references to array elements are called subscripts, and set apart with parentheses. For example P(1,5) references the fifth element of the first row of array P; 1 and 5 are the subscripts. In X(M,N), M and N are the subscripts.

The range of numbers that can be stored in a single element of an array, or in a simple variable is 1.0×10^{-99} to $9.99999999 \times 10^{99}$. There are nine digits of significance in this version of BASIC. There will be no radix or base conversion errors in this version of BASIC, because all numbers are stored internally in an exponential (floating point) base 10 format. All numbers are internally truncated to nine digits to fit this precision. Numbers may be entered and displayed in three formats: INTEGER, DECIMAL, and EXPONENTIAL. BASIC will automatically select the appropriate output format when executing a "PRINT" statement. Figure 3 identifies the three formats and gives examples.

INTEGER (NO DECIMAL POINT)
1 998 10 99999999
DECIMAL (FLOATING POINT NUMBERS)
.123456789 3.14 1.2E – 99
EXPONENTIAL (SCIENTIFIC NOTATION)
10E99 – 1.2345E76 1.2E – 99
E99 REPRESENTS 10^{99} AND E – 99 IS 10^{-99}
(E STANDS FOR EXPONENT)

Figure 3. Data Formats

Commands

Although most BASIC statements will have line numbers, it is possible to communicate in BASIC by typing certain types of statements without line numbers. This type of statements is called *commands.* A command is a BASIC statement that causes immediate action. In contrast, however, a typical BASIC program is entered into the computer by typing it a line at a time; later typing a command (RUN) to cause the computer to begin executing the program in accordance with the programming statements previously stored in the computer's memory. When BASIC is ready for you to type a command or is able to accept a programming statement for storage, the word "READY" is displayed on the terminal. After each such entry the system will prompt for an additional command or programming statement by displaying a "#". When a command is typed, the computer will take immediate action on that command, and when the command has completed it will again respond with the word "READY" and a "#".

This version of BASIC will understand seven different commands: NEW LIST RUN SAVE LOAD APPEND PATCH. Each of these seven commands are described in Table 2.

Command Name	Description of Command
NEW	The NEW command causes the current program to be erased. All working storage, internal pointers, and all variables will be reset. The effect of this command is to erase all traces of a program from memory and to start over.
LIST	The LIST command has several formats which can be used to cause all or selected portions of the current program to be displayed on your terminal. The lines are displayed in sequence with the lowest number first. The different LIST formats are as follows: LIST will list the entire program from start to finish.

LIST 100 will list only line 100.
LIST 100,200 will list lines 100 thru and including line 200.
LIST 100,1 will cause the program to begin listing at line 100 and will continue through to the end of the program. This operation will occur any time the second line number is less than the first.

RUN The RUN command causes the current program in memory to begin execution at the first statement number. RUN always starts with the lowest line number and will proceed in sequential order unless directed to do otherwise with a GOTO, GOSUB, ON, or IF statement.

SAVE The SAVE command causes the current program to be saved in a reloadable format on either the SWTPC AC-30 cassette interface or on a TELETYPE papertape punch, or an equivalent device. The control characters necessary to control the recording and playback mechanisms are output along with the program. More complete details are given in LOAD.

LOAD The LOAD command will first erase any program currently in working storage (as in the command NEW) and will then load a previously saved program. Control characters are output to control the reading mechanism. More complete details are given in APPEND.

APPEND The APPEND command functions exactly like the LOAD command except that memory is not cleared prior to the load function.

PATCH The PATCH command causes the computer to return to the MIKBUG operating system and outputs a carriage return, line feed, and an '*' on the terminal device. If no memory belonging to BASIC (addresses _____ to _____) and the program counter (addresses A048 and A049) are not changed, typing a "G" will restart BASIC with your program intact. The PATCH command may even be inserted as a programming statement with a line number in your program. When the PATCH statement is encountered, con-

trol is transferred to MIKBUG. Upon typing a "G", control will return back to the line that immediately follows the patch command.

Note: All seven of the control commands described above may also be used as programming statements, if the control statement is typed following a line number. As a BASIC programming statement their action will be suspended until the program executes them in sequence.

Caution: Using the NEW statement as an executable statement will cause your program to self-destruct!

Program Statements

Each program statement line begins with a line number which must be an integer between 1 and 9999. Statements may be entered in any order, but they will be executed in numerical order. Blanks, unless enclosed by quotation marks, are ignored. Program statements are limited to 72 characters including blanks. Nineteen types of program statements are allowed and are described in the following paragraphs. For your convenience the statements are listed in alphabetical order, rather than someone's "easy-to-learn" sequence. This will make it easier to use as a reference manual.

DATA The DATA Program statement causes data to be stored as part of a program. This data will be used by a READ statement. Data statements do not execute; they merely specify data. Multiple data items in a data statement must be separated by commas. Data statements may be placed anywhere in a program, and will be read in sequence as required. When the data is read in a program, the data will be read in sequence from the first to last data statements, and from left to right within each data statement. Each data item may be read only once unless a RESTORE statement is executed. BASIC keeps track of data with a *pointer.* When the first READ statement is encountered, the pointer indicates that the first data item in the first data statement is to be used; the pointer is then moved to the second item of data, and so on. An example of the DATA program statement format is as follows;

```
100 DATA 123, 34.5, 695, 500
```

DIM The DIM Program statement allocates memory space for an array. In this version of BASIC, one or two dimensioned arrays are allowed. Maximum array size is 255×255

elements. (This is not as much of a limitation as you might think—you would need 393,216 bytes of memory to contain an array of 65535 elements! and this doesn't allow for any memory to store the program code. . .) All array elements are set to zero by the DIM statement. Dynamic re-dimensioning of arrays is not allowed. Once an array has been defined, it must not be re-defined in the same program. If an array is not explicitly defined by a DIM statement, it is assumed to be defined either as an array of 10 elements or as an array of 10 × 10 upon the first reference to it in a program.

An array can be allocated only once in a given program, implicitly and explicitly. Only the variables A thru Z (not followed by a number) may be dimensioned. This does not prevent the use of a simple variable of the same name.

The working size of an array may be smaller than its physical size. For example, an array declared 5 × 5 in a DIM statement may be used to store fewer than 25 elements; the DIM statement supplies only an upper bound on the number of elements. In the example of A(5,5) the first position of this array is A(1,1) and the last is A(5,5). An example of the DIM Program statement format is as follows;

```
10 DIM A(5,5), Z(100,2)
```

END The END Program statement causes the program to stop executing. BASIC will print the word "READY" and a "#" on the terminal device indicating that it is now able to accept commands. In this version of BASIC, END may appear more than once and need not appear at all. It is recommended, however, that the last statement of a program be an END statement. This clearly shows that a program has been completely loaded or saved. An example of the END Program statement format is as follows;

```
9999 END
```

FOR The FOR Program statement allows repetition of a group of statements within a program for a specified number of times. The variable name that follows the word 'FOR' is used to identify the related NEXT statement. In addition the variable is initially set to the

value of the first expression (expression1). All statements between the FOR and the related NEXT statement are then executed. The named variable in the FOR statement is then incremented by the STEP value (expression3) and compared to the upper limit (expression2). If the increment creates a value that is *greater than* the upper limit (expression2) control of the computer is passed to the statement immediately following the NEXT statement. If the increment operation results in a sum that is equal to or less than the upper limit, the programming statements that fall between the FOR and NEXT statements will be repeated. This looping process will be continued until the upper limit has been exceeded, or as the result of an IF statement, control is passed to a statement outside the scope of the programming loop. If no STEP value is specified, a value of +1 is assumed. The loop will be executed once, regardless of the value of the variable. Although expressions are permitted for the initial, final, and step values in the FOR statement, they will be evaluated only once, the first time the loop is entered. It is not possible to use the same indexed variable in two loops if they are nested. When the statement after the NEXT statement is executed, the variable is equal to the value that caused the loop to terminate, and not the TO value itself. The step size need not be an integer. For instance,

```
100 FOR N = 1 TO 2 STEP .01
```

is a valid statement which will produce exactly 100 loop executions, incrementing N by .01 each time. A negative step size may also be used, as seen below:

```
100 FOR Q1 = 100 TO 50 STEP – 2.5
```

Examples of proper FOR Program statement formats are as follows;

```
320 FOR K = 1 TO 300
•
•
•
500 FOR variable = expression1 TO
expression2 STEP expression3
•
•
•
760 FOR L9 = 200 TO 1 STEP – 1
```

See the description of NEXT statement for additional information.

GOSUB The GOSUB program statement causes the program to transfer control to the subroutine at the specified line address and returns to the line immediately following GOSUB XXXX. A subroutine is a sequence of instructions which performs a task that has use in more than one place in a program. To be able to use such a sequence of programming statements, BASIC provides the facility (GOSUB) to call upon such a sequence from more than one place in the program.
The subroutine is actually a sequence of instructions that will receive control upon the execution of a GOSUB statement. Upon completion of the subroutine, control is returned back to the mainline of the program by execution of a return statement. The statement that immediately follows the original GOSUB statement will receive control when the RETURN is executed.
Subroutines may be nested to a level of 8 deep. That is, one subroutine may call a subroutine, which can call a subroutine,. . . to a nested depth of 8 subroutines. An example of the GOSUB program statement format is as follows;

```
235 GOSUB 9010
```

GOTO The GOTO Program statement transfers control of the computer to the statement number specified. Program control will continue sequentially from the new statement. An example of the GOTO program statement format is as follows;

```
100 GOTO 375
```

IF The IF Program statement transfers control to a specified statement if the result of a conditional comparison is true.
The IF statement is used to control the sequence of program statements to be executed, depending on specific conditions. If the relational expression is true, then control is given to the statement or statement number declared after the THEN. If the relational expression is false, program execution continues at the statement that immediately follows the IF statement. The

possible relational operators for use with the IF statement are as follows;

= Equal
<> Not equal
< Less than
> Greater than
<= Less than or equal
>= Greater than or equal

If a BASIC statement is specified after the THEN condition, rather than a statement number, the statement specified will be executed if the conditional expression is true, and control will be passed to the statement immediately following the IF statement.

It is possible to code another IF statement as the statement following the THEN condition—. This will give the equivalent of an "AND" condition.

When evaluating relational expressions, arithmetic operations take precedence in their usual order, and the relational operators are given equal weight and are evaluated last.

Examples of proper IF program statements formats are as follows;

```
190 IF X=56 THEN 300
•
•
•
200 IF Z9 D+Y-(7*S) THEN PRINT Z9;
"is out of range"
•
•
•
2022 IF expression1 relational-test ex-
pression2 THEN basic-statement
•
•
•
2022 IF expression1 relational-test ex-
pression2 THEN statement-number
```

An example of A program to illustrate IF statements is as follows:

```
10 INPUT X
20 IF X=1 THEN PRINT "EQUAL TO 1"
30 IF X<>1 THEN PRINT "NOT EQUAL
   TO 1"
40 IF X<1 THEN PRINT "LESS THAN
   1"
```

```
50 IF X>1 THEN PRINT "GREATER
   THAN 1"
60 IF X< = 1 THEN PRINT "LESS THAN
   OR EQUAL TO 1"
70 IF X> = 1 THEN PRINT "GREATER
   THAN OR EQUAL TO 1"
80 PRINT
90 GOTO 10
100 END
```

INPUT The INPUT program statement allows users to enter data from the terminal during program execution and to assign that value to a variable. The variable specified may be either a simple variable or a specified element in an array.

When the program comes to an INPUT statement, a question mark is displayed on the terminal. The user then types in the requested data separated by commas and followed by a carriage return. If no data is entered, or if the data entered is insufficient, the system prompts the user for more data with an additional question mark. Only numerical constants can be given in response to an input statement. Any number of variables may be specified on the INPUT statement, within the confines of the 72 character line, and the user must respond with a value for each specified variable. An example of the INPUT program statement format is as follows;

```
100 INPUT A,C,D
```

LET The LET program or assignment statement is used to assign or specify the value of a variable. The value may be an expression, a number, or another variable. The keyword LET is optional, and need not be specified. There are four functions which may be used inside a formula on a LET statement. ABS (absolute value), INT (integer value), RND (random number), and SGN (sign *not sine*) may be used inside formulas or LET statements or anywhere else that a mathematical expression is allowed. Examples of proper LET statement formats are as as follows;

```
100 X = 123.45
110 LET Z9 = INT(RND(0)*100)
120 T5 = (A*A)*3.14
```

```
•
•
300A(1,1) = X + P(M,N)
•
•
400Z(I) = X
```

NEXT The NEXT program statement is used to define the end of a "FOR. . . NEXT" loop. The variable specified in the NEXT statement identifies the associated FOR statement. Multiple FOR. . . NEXT statements may be used in the same program; They may also be nested (placed inside one another.) When nested, the innermost loop or loops must be wholly contained inside the outermost loop. The range must not overlap. An example of the NEXT program statement format is as follows;

220 NEXT K

Figure 4, illustrates proper and improper nesting.

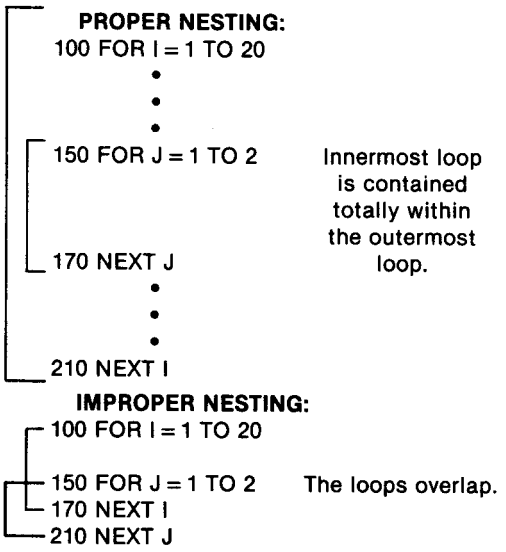

PROPER NESTING:
100 FOR I = 1 TO 20
150 FOR J = 1 TO 2
Innermost loop is contained totally within the outermost loop.
170 NEXT J
210 NEXT I
IMPROPER NESTING:
100 FOR I = 1 TO 20
150 FOR J = 1 TO 2
The loops overlap.
170 NEXT I
210 NEXT J

Figure 4. FOR. . . NEXT NESTING

For additional information see the description of the FOR statement.

ON

The ON program statement provides a computed GOTO or GOSUB mechanism. Instead of specifying the programming sequence of:

```
10 if X = 1 THEN 100
20 if X = 2 THEN 200
30 if X = 3 THEN 300
40 if X = 4 THEN 400
```

You could have used the following statement;

```
10 ON X GOTO 100,200,300,400
```

The computer would evaluate X, truncate the resulting value to an integer value, would then select the "Xth" statement number in the list of line numbers, and would use that value as part of a "GOTO" statement. Control of the computer would then be received by the selected line number.

In the example given above if the value of X was between 3 and 3.99999999 then the computer would GOTO 300. If the value of X was 1 then the computer would "GOTO 100". If the value of X is less than 1, or greater than the number of items in the list, the computer will generate an error message, and program execution will stop.

ON X *GOSUB* 100,200,300,400 works in a similar manner, except that the specified line numbers are assumed to be the start of a subroutine. When a RETURN statement has been executed, control on the computer will be returned to the statement immediately following the "ON X GOSUB" statement.

Note: The value X in the examples above can be any formula or mathematical expression.

(Additional note for FORTRAN Programmers or individuals trying to convert FORTRAN to BASIC):

The FORTRAN IF statement can be simulated in BASIC thru the use of the ON . . . GOTO statement.

FORTRAN: IF (X-Y)20,50,10

BASIC: ON SGN(X-Y)2 GOTO 20,50,10

PRINT

The PRINT program statement causes the values of any variables specified in the PRINT statement to be output or printed on the terminal device. The PRINT statement

can also be used to print out text, by enclosing the desired text to be printed in quotation marks ("). The statement

```
100 PRINT "HI THERE, I AM A COMPUTER"
```

would cause the text "HI THERE, I AM A COMPUTER" to be printed on the terminal device (the quotation marks would not be displayed or printed.)

The PRINT statement also makes use of the comma (,) and semicolon (;) to control the formatting of a print line.

In the statement PRINT A,B the numerical value of A will be printed beginning in the left hand margin, and the numerical value of B will be printed in column 18. Basic defines the 72 column line width into 4 zones starting in columns 18,36,48, and 60. The use of a comma in between data items will cause the printing mechanism to be advanced to the start of the next zone. This feature allows you to prepare tabular table data easily. The semicolon disables this "advance to next zone" mechanism and will cause the data items to be printed one after another. Numeric values will be separated by a blank for readability.

In addition if the PRINT statement is terminated by a semicolon, the computer will suppress the normal "carriage return, line feed", that is, automatically output at the end of each line. PRINT statement used with no operations or data items will cause a single blank line to be output.

This version of BASIC supports two special functions (TAB & CHR) that can be used only on the PRINT statement. The TAB function can be used to format items starting in columns other than 18,36,48, and 60. The CHR function is used to create special control characters that may not be generatable from your keyboard.

READ The READ program statement is similar in function to the INPUT statement except that the next DATA item is read, rather than getting a numeric value from the terminal. READ statements cause values in the DATA buffer to be accessed in a left to right, top to bottom sequential manner and assigned to

the variable named in the read statement. If more than one variable is named in the READ, the values read from the DATA statements will be assigned, in the order read to the variables named on the READ staement. For additional information, see the description of the DATA and RESTORE statements. Examples of proper READ Program statement formats are as follows;

```
100 READ A
200 READ a,q9,B
```

REM The REM program or remark statement allows insertion of a line of remarks or comments in the listing of a program. REM Lines are saved as a part of a BASIC Program, and are printed when the program is listed, but they are ignored when the program is executing. An example of the REM program statement is as follows;

```
300 REM
```

RESTORE In the discussion on the DATA and READ statements, it was stated that once a data item was read, it could not be re-read. This is only partially true. The RESTORE program statement is used to reset the internal data pointer that BASIC uses, back to the first data item on the first data statement. Thus the entire data list may be re-read from the beginning.

The only way that you can re-read a particular data item, is to keep track of how far down the data list it is; RESTORE the data pointer, and then READ "n-1" numbers throwing all of the previous numbers away. The data pointer has now been "reset" to a particular data item, so that it can be re-read. . . clearly not a very practical technique for most programs!!! An example of the RESTORE program statement format is as follows;

```
500 RESTORE
```

RETURN The RETURN Program statement is used to define the logical end of a subroutine, and when executed will cause control of the computer to be transferred to the statement immediately following the *last* GOSUB executed. A subroutine may call a subroutine, which calls still another subroutine. This

may be done to a nested level of 8 deep. You must take care not to execute a return statement unless you have executed a GOSUB first. An example of the RETURN Program statement format is as follows;

```
1000 RETURN
```

STOP The STOP program statement operates in a similar fashion to the END statement; both cause the currently running BASIC program to stop running. The STOP statement, however, will identify the line number of the statement that caused the stop; while the END statement merely stops the program without identifying where the "end" occurred. This feature of the STOP statement makes it useful in identifying "error" conditions. For instance, in the following example the computer will print "STOP at 100" if the number typed in response to the INPUT statement is less than zero.

```
90 INPUT X
100 IF X<0 THEN STOP
```

An example of the STOP Program statement format is as follows;

```
900 STOP
```

FUNCTIONS

The following functions are available in Robert Uiterwyk's 4K 6800 BASIC.

ABS(x) The ABS or absolute function will take the absolute value of the expression x. If the value of the expression is a negative number then the value return by the absolute function will be a positive value. A positive value will remain a positive value.

if Z9 is a −32 then the ABS(Z9) will be a +32

if Q5 = +6 then the formula ABS(Q5 + 3) will return a value of +9.

INT(x) The INT or integer function is used to truncate or "chop-off" any decimals in a floating point number. Positive values will be chopped off to the next lower integer, and negative numbers will be rounded up to the next lower value.

INT(9.765) will result in a value of 9.

INT(−10.2) will result in a value of −11.

RND(x) The RND or random number generator generates a pseudo-random number ranging

between 0.0 and 1.0. If the argument (x) is not equal to zero, it will be used as a new seed for the random number generator. The value returned as the result of specifying a new seed should be ignored. A new random number will be received when the argument specified is zero. In this version of BASIC it is not necessary to seed the generator in order to start a new random series of numbers at the start of a game.

SGN(x) The SGN Or sign function returns a −1, 0, or +1 value depending on the magnitude of the expression (x). If x is less than zero, the SGN will return a value of −1. If x is equal to zero, SGN will return a value of zero. If x is greater than zero, SGN will return a +1

if Z9 is a −45.987 then SGN(Z9) will be a −1

if Q is a +34.59 then SGN(Q) will be a +1

if R is a −10 and S is a −5 then SGN (R-2*S) will be 0.

CHR(x) The CHR function is used to convert the value of X (which must be between 2 and 255) to a single ASCII character. Thus CHR(16) will create a "CONTROL-P" and CHR(22) will create a "CONTROL-U". The CHR function may be used only on the PRINT statement and is useful for creating special control characters required for terminal control.

TAB(x) The TAB function can be used only in a PRINT statement. The value of X will cause the print mechanism to be positioned at column X. TAB(10) will cause a tab to column 10, TAB(50) will cause a tab to column 50. The value of the TAB function must be from 1 to 72.

USER The USER function is used to call a user defined assembly language subroutine.

OPERATORS

Symbols used to instruct the microcomputer to perform some operation are called operators. This version of BASIC includes arithmetic and relational operators.

ARITHMETIC OPERATORS—Five standard BASIC arithmetic operators are provided with this versin of BASIC. The arithmetic operators are as follows:

- Unary negation
* Multiplication
/ Division
+ Addition
- Subtraction

RELATIONAL OPERATORS—Standard BASIC relational operators are provided for comparing the values of integer expressions. The relational operators are as follows:

= Equal
<> Not equal
< Less than
> Greater than
<= Less than or equal
>= Greater than or equal

ERROR MESSAGES

The following error messages can be created by BASIC at any time:

ERROR CODE	MESSAGE DESCRIPTION
1	OVERSIZED VARIABLE. A TAB, CHR, ON or subscript value was greater than 255.
2	INPUT ERROR.
3	ILLEGAL CHARACTER OR VARIABLE. An illegal variable name of an invalid character was found during the evaluation of a mathematical expression.
4	NO ENDING " IN PRINT LITERAL. A missing quotation mark was detected in a print statement. There must be an even number of quotation marks on a print statement. The character (") cannot be output as a text character except by specifying CHR(34).
5	DIMENSIONING ERROR. Only two dimensions are allowed, and must be a value between 1 and 255. Once defined an array may not be defined again. Place all DIM statements at the start of the program, where they can not be accidentally executed a second time. All references to an array which is defined on a DIM statement must occur after the DIM statement has been executed.
6	ILLEGAL ARITHMETIC. The most common cause of "illegal arithmetic" is two arithmetic operators following each other, without a

numeric value separating them. "2 + −3" is illegal arithmetic.

7 LINE NUMBER NOT FOUND. The "GOTO", "GOSUB" or "IF...THEN" line number does not exist in the program. If necessary a REM statement may be used as a dummy line to resolve a missing line number.

8 DIVIDE BY ZERO ATTEMPTED. You just can't do it! If there is no way to prevent it, place an "IF" statement prior to the divide and branch around the divide statement if a divide by zero might occur.

9 EXCESSIVE SUBROUTINE NESTING. The maximum number of subroutines that may be nested is 8. Generally when this occurs it is because a GOSUB is accidentally being executed, or somehow a return statement has not been executed.

10 RETURN without PRIOR GOSUB. You executed a RETURN statement without first executing a GOSUB. This is a no-no.

11 ILLEGAL VARIABLE. The name for the variable is not correct. An array may only be named with a single alphabetic character. A(1) is correct, A5(1) is not. A simple variable may be named either with a single alphabetic character, OR a single alphabetic character followed by a single numeric character. A, B0, C1 are legal names. 1A, BB, C3A are not legal names. This version of BASIC does not support character string variables (A$, B$, etc.)

12 UNRECOGNIZABLE STATEMENT. This first word in the statement was not a recognizable COMMAND or BASIC statement name; nor was it a variable (simple or array) name followed by an equal sign (=). This latter case is known as an implied LET statement.

13 PARENTHESIS ERROR. Either you have too many opening parenthesis "(" or too few closing ones ")". There must be exactly the same number of opening and closing parenthesis in a programming statement.

14 MEMORY FULL. You just ran out of memory. Try to reduce the number of variables used in the program. If arrays are being used make

sure that they are dimensioned to the exact size required. Reduce the amount of remarks and the amount of PRINT statements used. Look and see if any routines are similar enough in nature to be made a subroutine. Look for more efficient algorithms to solve your problem. Split the program up into several different programs. If all else fails, go buy some more memory!

15 SUBSCRIPT ERROR. The array was defined as a one dimensional array and the reference contains two dimensions, or vice-versa.

16 EXCESSIVE FOR LOOPS ACTIVE. The maximum number of nested FOR...NEXT loops is 8.

17 NEXT WITHOUT CORRESPONDING FOR. A NEXT statement has been found without a corresponding FOR statement. The named variable that occurs after the word "FOR" and after the word "NEXT" is used to identify corresponding "FOR" and "NEXT" statements.

18 MISNESTED FOR NEXT LOOPS. The computer has detected an improper nesting for multiple "FOR...NEXT" statements. See Figure 3 for an explanation of the proper and improper way to nest "FOR...NEXT" statments.

19 READ STATEMENT ERROR. Read statement was executed, and there is not enough data in the data buffer to satisfy the READ request.

20 ERROR IN ON STATEMENT. The expression when truncated to an integer value resulted in either a value less than one, or in a value that is greater than the number of statement numbers in the list. In the example below an error 20 will occur if the value of (X-Y) is less than 1 or greater than 5.
100 ON(X-Y) GOTO 1000,235,3000,20,35

21 INPUT OVERFLOW. More than 72 characters were typed on the line.

GRAMMAR REFERENCE SUMMARY

Table 3, Quick definition of BASIC Statements, and Table 4, BASIC functions, provide a quick reference for using, Robert Uiterwyk's 4K 6800 BASIC Interpreter.

Table 3. Quick definition of BASIC Statements

STATEMENT	DEFINITION	RELATED STATEMENT(S)
DATA	'STORES' DATA TO BE USED BY READ STATEMENT	READ RESTORE
DIM	DEFINES SIZE OF ARRAYS (1 or 2 dimensions are allowed.)	
END	DEFINES END OF PROGRAM	STOP
FOR	DEFINES START OF PROGRAMMING LOOP	NEXT
GOSUB	TRANSFERS CONTROL TO A SUBROUTINE	RETURN
GOTO	TRANSFERS CONTROL TO OTHER THAN THE NEXT SEQUENTIAL STATEMENT	
IF	CONDITIONAL TEST USUALLY USED TO TRANSFER CONTROL "IF"	THEN
INPUT	GET DATE FROM KEYBOARD OF ATTACHED TERMINAL	
LET	ASSIGNMENT OF CALCULATION STATEMENT	
NEXT	DEFINES END OF PROGRAMMING LOOP	FOR
ON	COMPUTED "GOTO" OR "GOSUB"	GOSUB GOTO
PRINT	PUT DATA TO PRINTER OR CRT OF ATTACHED TERMINAL	
READ	GET DATA FROM "DATA" STATEMENTS	DATA RESTORE
REM	PROGRAMMER'S COMMENT	
RESTORE	RESETS THE DATA POINTER IN "DATA" STATEMENT	DATA READ
RETURN	DEFINES END OF SUBROUTINE — CAUSES CONTROL TO BE TRANSFERRED BACK TO STATEMENT IMMEDIATELY FOLLOWING THE GOSUB STATEMENT	GOSUB

STOP	DEFINES THE END OF THE PROGRAM: AND IDENTIFIES THE STATEMENT NUMBER OF THE STOP.	END

Table 4. BASIC Functions

ABS( )	ABSOLUTE VALUE
CHR( )	CONVERSION OF VALUE TO A SINGLE CHARACTER
INT( )	INTEGER VALUE (TRUNCATES DECIMAL VALUE)
RND( )	RANDOM NUMBER GENERATOR; RETURNS VALUE FROM .000001 to .999999
SGN( )	SIGN OF X; RETURNS A + 1, 0, or – 1
TAB( )	USED IN PRINT STATEMENT ONLY; CAUSES A TAB TO THE SPECIFIED COLUMN
USER( )	CAUSES A CALL TO A USER DEFINED ASSEMBLY LANGUAGE SUBROUTINE

Appendix A

General Software Index

Compiled by Jim Schreier

The software published in *Interface Age* between December 1976 and December 1978 is rich and varied. Historically, this time period marked the apex of "hobbyist" software and the dawning of "business" programs. Generally, the 1977 pages of *Interface Age* contained assembled programs for the SC/MP, 6800 and 8080. The 1978 edition published software for a wider variety of chips plus BASIC, FORTRAN and PILOT programs. This variety of subject matter will interest most microcomputer users as well as provide a permanent record of excellent software.

This compilation is divided into two sequential sections: Software by High Level Language and Software by CPU chips.

The High Level Languages are in alphabetical order by the name of the BASIC. Most users will want to see what programs were published for their specific microcomputer. The term "GENERAL BASIC" means the documentation failed to identify a specific BASIC dialect. Of course, there is no such thing as a "General BASIC," yet. It is possible to translate one BASIC to another given enough time.

Likewise, the assembled software is arranged by CPU chips in numeric order. Some of the most important published software programs appear in this section. Foremost is the publication in four parts of the Livermore BASIC Interpreter.

ALTAIR BASIC v 4.0 Building and land development program
BASIC listing November 1978 p 88

ALTAIR Extended BASIC v 3.2 Computer(ese) speech writer
BASIC listing January 1978 p 150

ALTAIR Extended disk BASIC v 4.0 Number base conversion
2 BASIC listings for disk and non-disk applications May 1977 p 116

ALTAIR 8K BASIC v 3.2 Star Lanes
Fixes to the computer game October 1977 p 8

ALTAIR 8K BASIC v 3.2 Star Lanes
Computer game requiring 7K bytes June 1977 p 137

APPLE BASIC APPLE Star-Trek
small BASIC listing May 1977 p 134

APPLE BASIC LOGAN
BASIC listing for logic circuit analysis May 1977

APPLE II BASIC Automated dress pattern
floppy ROM™-2 BASIC and assembled listings September 1978 p 78 and 80a

APPLE II BASIC TV pattern generator
BASIC listing August 1978 p 160

BASIC-E Membership file maintenance
illustrating random disk files February 1978 p 159

BURROUGHS B6700 BASIC Machine Menace
artificial intelligence game March 1977 p 22

CBASIC Personal management program
CBASIC compiler listing August 1978 p 79

COMPUCOLOR BASIC European Roulette in color—game
BASIC listing August 1978 p 57

FORTRAN IV M6800 cross assembler
2 pass cross assembler with symbol table October 1977 p 152

FORTRAN IV M6800 cross assembler
erratum December 1977 p 134

GENERAL BASIC Aquarium maintenance
calculates 11 items December 1977 p 66

GENERAL BASIC ASCII control character program
BASIC listing February 1978 p 8

GENERAL BASIC Base number conversion
converts base 2-8-10-16 November 1977 p 161

GENERAL BASIC Bi-lingual math tutoring
English and Spanish BASIC listings September 1978 p 131

GENERAL BASIC Biorhythm—plotting game
corrected listing December 1977 p 144

GENERAL BASIC Biorhythm—plotting game
2 listings with 1 designed for BASICs without SIN function
October 1977 p 142

GENERAL BASIC Circuit analysis
BASIC listing August 1978 p 118

GENERAL BASIC Financial analysis program
calculates annuity-lump sum and bonds March 1978 p 55

GENERAL BASIC Function approximation with 2 variables
BASIC listing October 1978 p 77

GENERAL BASIC Harmonic analysis of point sources
noise analysis program October 1977 p 54

GENERAL BASIC Job costing with random numbers
based on number of trials December 1977 p 86

GENERAL BASIC Local mean time
difference between GMT and T calculated August 1977 p 32

GENERAL BASIC Local sidereal time and date
6 various time/date calculations August 1977 p 33

GENERAL BASIC Pico fumi—game
BASIC listing December 1978 p 141

GENERAL BASIC Play number—child's game
BASIC listing September 1978 p 129

GENERAL BASIC Pot roast cooking time
a sample BASIC listing December 1977 p 33

GENERAL BASIC Preprocessor for structured programming
BASIC listing February 1978 p 85

GENERAL BASIC Stop TV display routine
BASIC listing June 1978 p 155

GENERAL BASIC Urine sugar and insulin curve plot
BASIC listing December 1978 p 73

HEWLETT PACKARD BASIC Viking uplink/downlink
communications between earth and Mars on Viking I August 1977 p 78

HONEYWELL 1108 Random number program for security combinations
calculates 3 number series December 1977 p 148

IMSAI BASIC Business risk simulation
BASIC listing January 1978 p 91

IMSAI BASIC Business risk simulation
BASIC listing January 1978 p 91

IMSAI 12K Extended BASIC v 4.0 Injun Poker—game
4K BASIC listing December 1977 p 162

MICROSOFT BASIC Recovery of crashed TARBELL tapes
BASIC listing April 1978 p 149

MITS 8K BASIC Azimuth by celestial observations
BASIC listing August 1978 p 76

MITS 8K BASIC Checkers
BASIC listing with graphics March 1977 p 136

MITS 8K BASIC v 3.1 Crazy ball
video game for Polymorphic VTI board August 1977 p 171

MITS BASIC v 3.1 Base number conversion
converts base 2-split octal-8-10-16 November 1977 p 163

MITS BASIC v 4.0 Bowling
A Bowling simulation for 4 players July 1977 p 170

MITS BASIC v 4.0 General Payroll
10 programs using random files June 1977 p 108

MITS BASIC v 4.0 Household finance system
includes monthly and yearly reports December 1977 p 46

MITS BASIC v 4.0 Tic-Tac-Toe
program allows computer to lose once in a while August 1977 p 170

MITS Disk BASIC v 4.0 Heart attack probability
BASIC listing July 1978 p 58

MITS Disk BASIC Motel's General Ledger
(a) floppy ROM™ at 1200 baud September 1977 p 32A
(b) 14 listings November 1977 p 57

MITS Disk Extended BASIC v 4.0 Solar eclipse prediction
12K BASIC listing August 1977 p 43

MITS Disk Extended BASIC v 3.4 The word processor
floppy ROM™ with source code January 1978 p 32A and 75

MSI Disk BASIC Full function mailing list system
floppy ROM™ at 300 baud and BASIC listings May 1978 p 59 and 64a

NORTH STAR Disk BASIC 1040 tax calculation program
based on California tax tables January 1978 p 158

NORTH STAR Disk BASIC Crazy ball—game
BASIC listing January 1978 p 170

NORTHSTAR Disk BASIC Random access name and address retrieval
2 BASIC listings August 1978 p 104

NORTH STAR Disk BASIC The tax man—game
BASIC listing February 1978 p 140

PDP-10 BASIC BASIC cross assembler for the 8080
BASIC listing February 1978 p 79

PDP-10 BASIC Data base management system
BASIC listing August 1978 p 113

PDP-10 BASIC Hexapod
Robotic walking program June 1977 p 50

PILOT Animal guessing—CIA game
PILOT Listing September 1978 p 66

PITTMAN'S 6800 TINY BASIC Shooting Starts
2K BASIC games listing April 1977 p 110

PROCESSOR TECHNOLOGY 5K BASIC Microcomputer Stock Options
4.2K BASIC listing designed for a stock market hedge February 1977 p 37

SWTPC 8K BASIC v 1.0 Calculating the position of Venus
calculates degrees or hours from the sun August 1977 p 55

SWTPC 8K BASIC v 1.0 Checkbook balancing
BASIC listing May 1977 p 126

SWTPC 8K BASIC v 1.0 Day of the Week
BASIC listing from 1752 June 1977 p 128

SWTPC 8K BASIC v 1.0 Depreciation schedule analysis
BASIC listing of 4 types of depreciation September 1977 p 148

SWTPC 8K BASIC v 2.0 How to buy an apartment building
6 calculations supported January 1978 p 99

SWTPC 4K BASIC v 2.0 Resistor sorting
BASIC listing April 1978 p 167

SWTPC 4K BASIC v 2.0 Sine-Cosine-Exponential-Logarithm-Square Root-
Algorithms February 1977 p 103

SWTPC 8K BASIC v 2.0 World power—game
BASIC listing February 1978 p 171

SWTPC 8K BASIC v 2.0 Zip code sort and print program
BASIC listing April 1978 p 91

TDL 8K BASIC Personal accounts payable
tape based program December 1977 p 59

2650 Interface design
5 assembled listings November 1977 p 87

6502 Robot control routines
assembled initialization program-speed control-joystick control-etc.
April 1977 p 24

6502 Search subroutine for the 6502 Disassembler
assembled listing: See *Interface Age* September 1976 p 14
October 1977 p 146

6502 Sidereal/Solar clock for KIM-1
assembled listing August 1977 p 37

6800 A better memory test
assembled listing with object code July 1977 p 160

6800 A faster TTY paper tape load and dump
2 assembled listings with object code September 1977 p 162

6800 Bubble sort routine
assembled listing with object code September 1977 p 156

6800 Darkroom prompter program
assembled listing October 1978 p 84

6800 EXMON—Extended monitor system
assembled listing with object code April 1977 p 115

6800 GP monitor
assembled 4K RAM listing plus object code for 4-8-16 and 32K
October 1978 p 133

6800 High density tape load
assembled source listing with object code August 1977 p 167

6800 High density tape punch
assembled source listing with object code August 1977 p 163

6800 IAPS™ International ASCII Publication Standard conversion
assembled listings for 3 versions May 1978 p 73

6800 MIKBUG™ Do Nothing—nonsense game
object code listing December 1977 p 111

6800 PIA test
assembled listing without comments with object code July 1977 p 162

6800 Program data or address locator
assembled listing with object code dump December 1978 p 137

6800 Prototype board monitor program with PROM burner
assembled listings February 1977 p 115

6800 Re-entrant self-relative subroutine ROM
assembled listing March 1977 p 129

6800 Relocator
assembled listing March 1978 p 166

6800 Software aid for firmware production
2 assembled listings March 1978 p 157

6800 Stars—a game
assembled listing with object code October 1977 p 161

6800 Text Editor
assembled listing December 1976 p 89

6800 Uiterwyk's 4K BASIC Interpreter
floppy ROM™ at 300 baud disk-object code listing with assembled binary loader May 1977 p 33

6800 Uiterwyk's 4K BASIC Interpreter: 3 routines
assembled MIKBUG™ jumps-ACIA break routine-biload command
July 1977 p 36

8080 2707 EPROM programming software
assembled listing December 1978 p 104

8080 A tabcounter for your edit buffer
assembled listing August 1978 p 151

8080 An 8080 binary tape monitor
assembled listing February 1978 p 153

8080 An Intel hex-format paper tape monitor
assembled listing March 1978 p 162

8080 Blockade—VDM-1 video game
assembled listing with object code November 1977 p 167

8080 Cassette Operating System
assembled subroutine listings April 1977 p 129

8080 Convert Motorola 6800 hex format to Intel format
assembled listing with object code May 1977 p 109

8080 Conway's Game of Life
assembled listing for Processor Technology's video display board
May 1977 p 138

8080 Copy/Sort/Search data manipulation
assembled listing September 1978 p 136

8080 Cromemco Dazzler graphics interface driver
assembled listing January 1978 p 153

8080 Diablo printer drive routine
assembled listing July 1977 p 25

8080 Dr. Wang's tiny BASIC
assembled listing December 1976 p 89

8080 Generalized string sorting routine
BASIC and 6 assembled listings November 1978 p 131

8080 I/O driver for PERSCI 1070 intelligent disk controller
assembled listing September 1977 p 153

8080 IAPS™International ASCII Publication Standard conversion
asembled listing May 1978 p 79

8080 IAPS™ International ASCII Publication Standard conversion
assembled load-create-verify-dump listings June 1978 p 148

8080 Intel hex format paper-tape punch program
assembled listing with object code July 1977 p 155

8080 Intelligent terminal program
assembled listing with object code September 1977 p 75

8080 Interval timer design
5 assembled routines January 1978 p 127

8080 Intrupped driven floppy disk controller for the S-100 bus
assembled listing May 1978 p 152

8080 Livermore BASIC Interpreter Part I
BASIC and assembled plot functions December 1976 p 115

8080 Livermore BASIC Interpreter Part II
assembled listing January 1977 p 97

8080 Livermore BASIC Interpreter Part III
assembled floating point math package listing February 1977 p 104

8080 Livermore BASIC Interpreter Part IV
assembled octal debugging listing March 1977 p 121

8080 Look—byte look-up program
assembled listing May 1978 p 167

8080 Memory catalog program
assembled listing May 1978 p 170

8080 Memory object code search routine
assembled routine February 1977 p 121

8080 Monitor initialization and printer control
2 assembled Daisywheel listings October 1978 p 111

8080 Morse Code generator
assembled listing October 1978 p 89

8080 Octal Monitor Program
assembled listing February 1977 p 13

8080 Piranha—game
assembled listing with object code December 1977 p 166

8080 Polymorphic Ideaboard software
2 assembled routines May 1977 p 98

8080 Punch and read Intel formatted tape
assembled listing December 1977 p 152

8080 Random Number Generator Program
assembled listing with a BASIC Chi-square subroutine February 1977 p 100

8080 Robot's random programming approach
assembled listing April 1978 p 158

8080 Text editor for XEK and PTCo Assemblers
assembled listing October 1978 p 140

8080 Video Chase—a game
assembled listing with object code October 1977 p 167

8080A Resident Operating System
1K SOL assembled listing January 1977 p 90

9900 Interactive monitor
assembled listing July 1978 p 163

MOLYPROCESSOR Molyprocessor Music
assembled listing November 1977 p 153

SC/MP CRT Interface firmare
assembled listing January 1978 p 120

SC/MP Lock combinations program
assembled listing July 1977 p 44

SC/MP NIBL—Extended tiny BASIC
assembled listing January 1977 p 113

SC/MP Seiko printer interface and program
assembled listing May 1977 p 124

SC/MP Word G—a word game
assembled listing December 1977 p 136

Z-80 BYTEMOVER™ for Cromemco's BYTESAVER™
assembler listing January 1977 p 70

Z-80 Date and time for CP/M DOS
assembled listing August 1978 p 154

Z-80 Disk I/O keyboard handler
assembled listing September 1977 p 167

Z-80 Letter writing program
assembled listing September 1978 p 94

Z-80 Patches to the MITS 12K Extended BASIC
assembled routines March 1977 p 138

Z-80 Spaceship simulator
assembled service routine March 1978 p 74

Z-80 TTY handler for the Z-80 development system
serial and test assembled listings May 1977 p 112

Z-80 User I/O for LIFEBOAT CP/M™ DOS
assembled listing December 1978 p 133

Appendix B

Available Back Issues

April 1976	2.25
October 1976	2.25
November 1976	2.25
March 1977	2.25
May 1977	
June 1977	2.50
July 1977	2.50
August 1977	2.50
September 1977	2.50
October 1977	2.50
November 1977	2.50
February 1978	2.75
April 1978	2.75
July 1978	2.75
August 1978	2.75
September 1978	2.75
October 1978	2.75
November 1978	2.75
December 1978	2.75

March 1979	2.75
April 1979	2.75
May 1979	2.75
June 1979	2.75
July 1979	2.75
August/September 1979	2.75

INTERFACE AGE MAGAZINE Dept. BI P.O. Box 1234,
Cerritos, CA 90701

Name (print) _____

Address _____

City_____ State_____ Zip_____

Price includes 50¢ for postage and handling.

VISA# _____ M/C# _____

Exp. Date _____ Signature _____

Index

APPEND, 283

Bagels, 189
BASIC, 1, 139, 229, 278
BREAK, 183

CHKIO, 167
CHKSGN, 161
CKHLDE, 161
CLEAR, 184
Colons, 188
Commands, 282

DATA, 284
Data formats, 281
DIM, 284
DIRECT, 149
DIVIDE, 161
DO, 185

END, 188, 285
ENDCHK, 162
ERROR, 162
EXEC, 149
EXPR, 157
External Routine Calls, 245

FIN, 162
Floppy ROM, 229
FNDLN, 163
FOR, 154, 186, 285
FUNCTIONS, 294

GETLN, 163
GOSUB, 153, 183, 287
GOTO, 183, 287

IF, 287
IFF, 156
IF/THEN, 184
INP, 160
Input, 8
INPUT, 154, 186, 289
Interpreter, 8

Kansas City Standard, 230

LET, 156, 289
LIST, 153, 183, 188, 282
LOAD, 283

MAIN, 147
Math operator, 2
Memory map, 238
MIKBUG, 233
MVDOWN, 166
MVUP, 166

NEXT, 154, 186, 290
NEW, 184, 282
NIBL, 173

ON, 291
ONION, 174
OPERATORS, 295

OUT, 160
OUTC, 167

PAGE, 184
PATCH, 283
PEEK, 160, 182
POKE, 160, 182
POPA, 166
PRINT, 153, 186, 291
Program control, 183, 284
PRTLN, 164
PRTNUM, 164
PRTSTG, 164
PUSHA, 166

QTSTG, 164

READ, 262
REM, 156, 188, 293
RESTORE, 293
RETURN, 153, 184, 293
ROM, 229, 240
RUN, 183, 283

SAVE, 283
SETVAL, 162
Software, 301
STAT, 184
STOP, 294
SUBDE, 161

TABLES, 149
Tiny BASIC, 140, 146
Trap, 22
Truth, 191
UNTIL, 185
USR, 160

Variable names, 281

WAIT, 160

ISBN: 0-918398-36-3